IFIP Networking 2016

May 17-19, 2016
Vienna, Austria

Proceedings

IFIP Networking 2016 is technically co-sponsored by as well as hosted by:

IFIP Networking 2016 gratefully acknowledges the generous support from the following patrons:

Proceedings: Patrick Poullie
Cover/preface: Burkhard Stiller

Organizing Committee

General Chair

Peter Reichl, University of Vienna, Austria

Technical Program Chairs

Jörg Ott, Technische Universität München, Germany

Christos Papadopoulos, Colorado State University, USA

Fabio Ricciato, University of Ljubljana, Slovenia

Steering Committee

Jordi Domingo-Pascual, Universitat Politècnica de Catalunya (UPC), Spain (Chair)

Andrea Passarella, IIT-CNR Pisa, Italy

Aiko Pras, University of Twente, The Netherlands

Henning Schulzrinne, Columbia University, USA

Jozef Wozniak, Gdansk University of Technology, Poland

Publication Chairs

Patrick Poullie, University of Zürich, Switzerland

Burkhard Stiller, University of Zürich, Switzerland

Publicity Chairs

Tobias Hoßfeld, University of Duisburg-Essen, Germany

Aruna Balasubramanian, Stony Brook University, USA

Local Arrangements

COSY Research Group, University of Vienna, Austria

Web Responsible

Hannes Weisgrab, University of Vienna, Austria

Technical Program Committee

Rui Aguiar, University of Aveiro, Portugal
Kevin Almeroth, University of California, Santa Barbara, USA
Nils Aschenbruck, University of Osnabrück, Germany
Marinho Barcellos, Federal University of Rio Grande do Sul, Brazil

Francesco Gringoli, University of Brescia, Italy
Andrei Gurtov, Aalto University, Finland
David Hausheer, TU Darmstadt, Germany
Boudewijn Haverkort, University of Twente, The Netherlands
Markus Hofmann, Bell Labs/Alcatel-Lucent, USA
Chengchen Hu, Xi'an Jiaotong University, P.R. China
Longbo Huang, Tsinghua University, P.R. China
Karin Hummel, ETH Zurich, Switzerland
David Hutchison, Lancaster University, United Kingdom
Esa Hyytiä, Aalto University, Finland
Paola Iovanna, Ericsson, Italy
Hongbo Jiang, Huazhong University of Science and Technology, P.R. China
Yuming Jiang, Norwegian University of Science and Technology (NTNU), Norway
Rahim Kacimi, University of Toulouse, France
Gunnar Karlsson, KTH Royal Institute of Technology, Sweden
Andreas J. Kassler, Karlstad University, Sweden
Hyun-chul Kim, Sangmyung University, Korea
Kimon Kontovasilis, NCSR Demokritos, Greece
Dimitrios Koutsonikolas, University at Buffalo, SUNY, USA
Udo R. Krieger, Otto-Friedrich-University Bamberg, Germany
Fernando Kuipers, Delft University of Technology, The Netherlands
Thomas Kunz, Carleton University, Canada
Axel Küpper, TU Berlin, Germany
Xavier Lagrange, Institut Mines Telecom / Telecom Bretagne, France
Guy Leduc, University of Liege, Belgium
Kenji Leibnitz, NICT, Japan
Jorg Liebeherr, University of Toronto, Canada
Alex Liu, Michigan State University, USA
Bin Liu, Tsinghua University, P. R. China
Jaime Lloret, Universidad Politecnica de Valencia, Spain
Renato Lo Cigno, University of Trento, Italy
Olaf Maennel, Tallinn University of Technology, Estonia
Dario Maggiorini, University of Milano, Italy
Zoubir Mammeri, Paul Sabatier University, France
Mahesh Marina, University of Edinburgh, United Kingdom
Ivan Martinovic, University of Oxford, United Kingdom
Maja Matijasevic, University of Zagreb, Croatia
Deep Medhi, University of Missouri-Kansas City, USA
Michael Menth, University of Tuebingen, Germany
Sándor Molnár, Budapest University of Technology and Economics, Hungary
Edmundo Monteiro, University of Coimbra, Portugal
Luis Muñoz, University of Cantabria, Spain
Maurizio Naldi, University of Rome "Tor Vergata", Italy
Ilkka Norros, VTT Technical Research Centre of Finland, Finland
Philippe Owezarski, LAAS-CNRS, France
Elena Pagani, University of Milano, Italy
Ai-Chun Pang, National Taiwan University, Taiwan
Andrea Passarella, IIT-CNR, Italy
Veljko Pejovic, University of Ljubljana, Slovenia
Colin Perkins, University of Glasgow, United Kingdom
Harry Perros, North Carolina State University, USA
Antonio Pescapé, University of Napoli Federico II, Italy

Dirk Pesch, Cork Institute of Technology, Ireland
Thomas Plagemann, University of Oslo, Norway
George Polyzos, Athens University of Economics and Business, Greece
Ana Pont, Universitat Politécnica de València, Spain
Ioannis Psaras, University College London, United Kingdom
Guy Pujolle, University Pierre et Marie Curie - Paris 6, France
James Roberts, IRT SystemX, France
Jean-Louis Rougier, TELECOM ParisTech / LTCI, France
Gerardo Rubino, Inria/Irisa, France
Stefano Salsano, University of Rome "Tor Vergata", Italy
Ricardo Schmidt, University of Twente, The Netherlands
Jens Schmitt, University of Kaiserslautern, Germany
Jürgen Schönwälder, Jacobs University Bremen, Germany
Stefano Secci, University Pierre et Marie Curie - Paris 6, France
Aruna Seneviratne, University of New South Wales, Australia
Christoph Sommer, University of Paderborn, Germany
Yang Song, IBM Research, USA
Otto Spaniol, RWTH Aachen University, Germany
Ioannis Stavrakakis, National and Kapodistrian University of Athens, Greece
Moritz Steiner, Bell Labs / Alcatel-Lucent, USA
Burkhard Stiller, University of Zürich, Switzerland
Aaron Striegel, University of Notre Dame, USA
Violet Syrotiuk, Arizona State University, USA
Yutaka Takahashi, Kyoto University, Japan
Y. C. Tay, National University of Singapore, Singapore
Chen Tian, Huazhong University of Science and Technology, P. R. China
Joe Touch, USC/ISI, USA
Gareth Tyson, Queen Mary, University of London, United Kingdom
Steve Uhlig, UK, United Kingdom
Piet Van Mieghem, Delft University of Technology, The Netherlands
Sandrine Vaton, Telecom Bretagne, France
Bing Wang, University of Connecticut, USA
Ning Wang, University of Surrey, United Kingdom
Cedric Westphal, Huawei Innovation Center, USA
Joerg Widmer, IMDEA Networks Institute, Spain
Sabine Wittevrongel, Ghent University, Belgium
Lars Wolf, Technische Universität Braunschweig, Germany
Tilman Wolf, University of Massachusetts, USA
Jozef Wozniak, Gdansk University of Technology, Poland
Fan Wu, Shanghai Jiao Tong University, P.R. China
Kui Wu, University of Victoria, Canada
Haiyong Xie, University of Science and Technology of China, P.R. China
Yang Xu, New York University, USA
George Xylomenos, Athens University of Economics and Business, Greece
Marco Zennaro, The Abdus Salam International Centre for Theoretical Physics (ICTP), Italy
Chi Zhang, University of Science of Technology of China, P.R. China
Zhi-Li Zhang, University of Minnesota, USA
Haojin Zhu, Shanghai Jiao Tong University, P.R. China

Table of Content

IFIP Networking 2016

P2P and Content Distribution

Transport Layer

Information-Centric Networking 2

Network Economics

Video Streaming

Workshop on Internet of People (IoP-W 2016)

Social Aspects of IoP

Analysis toward Supporting IoP System Design

Consistent Updates in Software Defined Networks: On Dependencies, Loop Freedom, and Blackholes

Klaus-Tycho Förster
ETH Zurich
foklaus@ethz.ch

Ratul Mahajan
Microsoft Research
ratul@microsoft.com

Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch

***Abstract*—We consider the problem of finding efficient methods to update forwarding rules in Software Defined Networks (SDNs). While the original and updated set of rules might both be consistent, disseminating the rule updates is an inherently asynchronous process, resulting in potentially inconsistent states. We highlight the inherent trade-off between the strength of the consistency property and the dependencies it imposes among rule updates at different switches; these dependencies fundamentally limit how quickly the SDN can be updated. Additionally, we consider the impact of resource constraints and show that fast blackhole free migration of rules with memory limits is NP-hard for the controller. For the basic consistency property of loop freedom, we prove that maximizing the number of loop free update rules is NP-hard for interval-based routing and longest-prefix matching. We also consider the basic case of just one destination in the network and show that the greedy approach can be nearly arbitrarily bad. However, minimizing the number of not updated rules can be approximated well for destination-based routing. For applying all updates, we develop an update algorithm that has a provably minimal dependency structure. We also sketch a general architecture for consistent updates that separates the twin concerns of consistency and efficiency, and lastly, evaluate our algorithm on ISP topologies.**

I. Introduction

The Internet as a whole is a wild place, full of autonomous participants. As such, it is naturally difficult to control centrally; instead, routing and congestion control is achieved through a selection of distributed protocols such as BGP and TCP. However, distributed protocols degrade performance, BGP cannot find the least congested path, and TCP will only crudely approximate the available bandwidth on the path selected by BGP. As a result, a loss of performance is to be expected and accepted. Many desirable properties such as drop freedom of packets, good utilization of links, or packet coherence are not as important as robustness. In contrast, individual networks that make up the Internet are controlled by single administrative entites. These include enterprise networks, ISP networks, data center networks, and wide area networks that connect the data centers of large organizations. The owners of these networks want to get the maximum out of their massive financial investment, which often runs into hundreds of millions of dollars per year (amortized). Towards this end, they have started replacing inefficient distributed protocols.

The technological driver to this paradigm shift are so-called Software Defined Networks (SDNs): In an SDN, the data plane is separated from the control plane, allowing the decision of where and how much data is sent to be made independent of the system that forwards the traffic itself. A centralized controller monitors the current state of the network, then calculates a new set of forwarding rules, and distributes them to the routers and switches [1], [2], [3], [4].

Are centrally controlled SDNs the beginning of the end of distributed protocols? Not so fast! After all, the central SDN controller has to inform the switches about updates, and a network is an inherently asynchronous place, where nodes might even be temporarily not accessible to the controller [4]!

In this paper we will discuss the problems that arise when updating rules in an asynchronous SDN-based network. We will show that despite the central control, distributed computing will have an important role, depending on the kind of consistency model one expects from the network. One of the most basic consistency properties is that packets should not loop. As a result, this property, which we call "loop freedom," is the starting point of our discussion. We will then discuss the broader space of consistency properties and highlight the inherent trade-off between the strength of the property and the intricacy of dependencies it induces among the actions of different switches. These dependencies fundamentally limit how quickly the SDN can be updated.

We build on our prior work [5], which showed that single-destination networks can be updated loop free in a distributed fashion, but did not consider the inherent computational complexity or dynamic architectures. We also extend the view on the consistency space, especially regarding blackholes.

We start in Section III by formally modeling consistent single-/multi-destination network updates, and show that not all updates can be sent out in one flush. In Section IV, we follow up by studying the NP-hardness of loop free updates. In Section V, we study maximizing the number of sent out updates at once and how to build a minimal dependency structure for applying all updates. Afterwards, in Section VI, we reveal the trade-off between consistency properties and update dependencies. Additionally, we consider the impact of resource constraints and show that fast blackhole free migration of rules with memory limits, i.e., a packet arriving at a switch must always have a matching rule to handle it, is NP-hard. We sketch a general architecture for consistent network updates in Section VII and conclude with Section VIII, where we present practical evaluation results.

ISBN 978-3-901882-83-8

II. Background and Related Work

From early papers on the topic (e.g., [6], [7]), we can learn that the primary promises of SDNs were that i) centralized control plane computation can eliminate the ill-effects of distributed computation (e.g., looping packets), and ii) separating control and data planes simplifies the task of configuring the data plane in a manner that satisfies diverse policy concerns. For example, to eliminate oscillations and loops that can occur in certain iBGP architectures, the Routing Control Platform (RCP) [6], [7] proposed a centralized control plane architecture that directly configured the data plane of routers in an autonomous system. However, as we gain more experience with this paradigm, a nuanced story is emerging. Even with SDNs, packets can take paths that violate policy [8] and traffic greater than capacity can arrive at links [3]. What explains this gap between the promise and these inconsistencies? The root cause is that promises apply to the eventual behavior of the network, *after* the data plane state has been changed, but inconsistencies emerge *during* data plane state changes.

Recent works have tackled specific pieces of this consistent update problem. Reitblatt et al. [8], [9] propose a per-packet consistency solution that we call "packet coherence"—each packet is routed entirely using the old rules or the new rules, and never a mix of the two sets; Katta et al. [10] propose extensions to this solution to reduce switch memory overhead. SWAN [3] and [11], [12], [13] propose solutions to ensure that link capacity is not exceeded. The work of Moses et al. [14] discusses balancing update performance versus periods of inconsistencies in a time-based update approach.

We make two contributions to this nascent line of work. First, beyond looking at consistency properties in isolation, we outline the broader consistency space and the fundamental hardness of ensuring different consistency properties. This perspective helps uncover the trade-off between the strength of the consistency property and the difficulty of ensuring it. Second, we investigate in detail loop freedom, a property that has not been considered despite being basic, except for the recent parallel work of [15], [16], [17]. The packet stamping solution of Reitblatt et al. [8] can ensure loop freedom by adding version numbers to packets, but because it ensures the much stronger property of packet coherence, it is slow and has high memory overhead. The whole network needs to be updated first, before being able to use the system—a long delay in updating single node induces a long delay for the complete network. Further, despite the extensions of Katta et al. [10], which trade-off switch memory for speed, packet stamping has high memory overhead because it simultaneously stores both old and new rules. Switch memory is a scarce commodity, with even future generations of switches reaching their memory limit easily when optimizing the network [3]. Our solutions, designed specifically for loop freedom, are faster and memory efficient. Interestingly, a majority of the motivating examples in [8] do not need packet coherence, only loop freedom.

Francois et al. [18],[19] consider avoiding transient loops during the convergence of link-state routing protocols. They argue that, due to high reliability requirements nowadays, one should try to avoid all packet losses. For the case of single-destination rules, they consider the routing tree T of the destination, layered into ranks equivalent to the depth. The ranks are then updated after another, causing depth(T) updates in total. Their mechanism design can achieve fast convergence even in tier-1 ISPs and is carefully fine-tuned for practical deployment [20]. Our work allows for updating nodes from different ranks in one update. As such, our number of updates is not linked to the maximum chain length in the tree, but rather on the maximum chain length in the dependencies imposed by the update in general.

Finally, Vanbever et al. [21] work on a related problem, and study the migration of a conventional (non-SDN) network to a new IGP protocol. The main differences to our work arise from the fact that they impose two restrictions on their model: First, every node must update all its rules at once. Second, only a single node may be updated at a time, one after another. In contrast, we can update individual forwarding entries for many nodes in parallel.

III. Model for Loop Free Routing Updates

We model a network as a set of connected routers and switches (from now on, *nodes*). Packets must be forwarded to their destination without loops. More formally, a network is a directed multi-graph with a set of nodes V, a set of destinations $D \subseteq V$, and a set of destination-labeled edges s.t. all edges labeled with the same set of destinations will not contain a directed loop. These edges form a directed spanning tree with d being the root and all edges being oriented towards d.

Definition 1: Let $T_d = (V, E_d)$ be a directed graph with V being the set of nodes, $d \in D$ being the sole destination, and E_d being the set of edges each labeled with d. The edge from $u \in V$ to $v \in V$ for destination d is noted as $(u, v)_d$. The labeled directed graph T_d is a *single-destination network*, if T_d is a spanning tree with all directed edges being oriented towards d.

Definition 2: Let V be a set of nodes and $D \subseteq V$ be a set of destinations. For all $d \in D$, let $T_d = (V, E_d)$ be a single-destination network and let $E_D = \bigcup_{d \in D} E_d$. Then the labeled directed multi-graph $T_D = (V, E_D)$ is a *multi-destination network*.

When a network needs to be updated, some (potentially all) nodes receive a new set of forwarding rules, leaving the network in a sort of limbo state. At some point all nodes will be updated, but until then, the network might not be consistent, i.e., it might induce loops.

Definition 3: Let $T_D^{old} = (V, E_D^{old})$ and $T_D^{new} = (V, E_D^{new})$ be multi-destination networks for the same set of nodes V and destinations D. Then $U_D = (V, E_D^{old}, E_D^{new})$ is called a *multi-destination network update*. If the labeled directed multi-graph $T_D = (V, E_D^{old} \cup E_D^{new})$ does not contain any loops of edges with the same label, then the update U_D is called *consistent* or *loop free*. A *single-destination network update* U_d can be defined analogously.

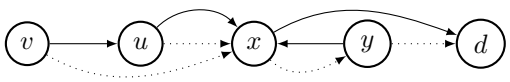


Fig. 1. Illustrating loop freedom. Not all updates can be sent out at once. Dotted edges are *new*, solid edges are *old*.

For an introductory example, consider the five-node single-destination network in Figure 1. Assume that we want to update the routing to destination d from the *old* pattern (solid edges) to the *new* pattern (dotted edges). A naïve method is to send out all updates (e.g., ask v to send packets destined to d to x) in one shot. However, during application of these updates, it might happen that x updates its rule before y, introducing a routing loop between x and y. This loop will eventually disappear, once y updates its rule, but in an asynchronous system with possible message delays and losses, we cannot guarantee when this will happen. Asynchronicity is not a technicality, as nodes in a production network can often react slowly (some switches might take up to $100\times$ longer than average to update [12]), or may not be accessible for some time to the controller [4]. Thus, solutions in which the network can quickly start using as many of the new rules as possible, while maintaining the consistency properties, are preferable.

IV. Updates and Default Rules

Interval routing and longest-prefix matching are common routing techniques for large networks. In interval routing (introduced in [22], cf. [23]), destinations $\{d_1, \ldots, d_{|D|}\}$ are ordered cyclically, and forwarding rules for a node are defined as disjoint intervals over the destinations, cf. [24], [25], [26]. In contrast, longest-prefix routing defines forwarding rules via prefixes of the destination IDs, which may overlap: If two rules are in conflict, the one with the longer matching prefix is chosen, cf. [27], [28]. Both techniques have great practical advantages, since multi-destination routing does not scale well: Even when considering just IPv4 (and not IPv6), no router on the market could store an individual rule for every IP–address. Furthermore, this fine–grained information is not available, since the complete knowledge over a network is usually restrained to one's own Autonomous System.

A subset of both techniques is multi-destination routing with the possibility of default routes. Nodes can either have individual forwarding rules for each destination or a default rule, cf. [29], i.e., all packets go to a specific other node (except for those that reached their destination at the current node). In this section, we show that maximizing a loop free update with default rules is an NP-hard problem – and therefore also NP-hard for both supersets.

Definition 4: Let $T_D = (V, E_D)$ be a multi-destination network and let $u, v \in V$. If all outgoing edges from u point at v in E_D, then those edges E_u may be merged into a *default edge*, labeled with all labels from D (but packets for a destination u do not get forwarded from u). We denote such an edge with $(u, v)_\forall$. I.e., we remove E_u from E_D and add $\{(u, v)_\forall\}$. Let the resulting set of edges of this iterated process be $E_{D,\forall}$. We call $T_{D,\forall} = (V, E_{D,\forall})$ a multi-destination network with default routes or *multi-default network*.

Definition 5: Let $T^{old}_{D,\forall} = (V, E^{old}_{D,\forall})$ and $T^{new}_{D,\forall} = (V, E^{new}_{D,\forall})$ be multi-default networks for the same set of nodes V and destinations D. Then $U_{D,\forall} = (V, E^{old}_{D,\forall}, E^{new}_{D,\forall})$ is called a *multi-default network update*. If the labeled directed multi-graph $T_{D,\forall} = (V, E^{old}_{D,\forall} \cup E^{new}_{D,\forall})$ does not contain any loops of edges with the same label, then the update $U_{D,\forall}$ is called *consistent* or *loop free*.

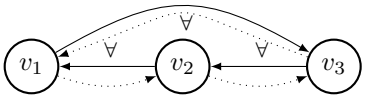


Fig. 2. Illustrating circular dependencies with default routes. Note that both in the old and new rules, no packet will loop: E.g., in the old rules, a packet sent out from v_1 will be forwarded to v_3, and possibly to v_2, but never to v_1 again - as all possible destinations were already reached on the path.

Let us start with an example of just three nodes in Figure 2. We want to update the three old default edges (drawn solid) to the three new default edges (drawn dotted). However, due to circular dependencies, not even a single edge can be updated without causing a loop. This problem can be handled by relaxing the constraints of default routing: One can prevent loops by breaking a single (default) rule into one helper rule for each of the two other destinations, introducing these rules during the update process and then removing them later. In general, this is not desirable, as memory constraints on routers can easily prevent introducing these additional helper rules, cf. [3]. Nonetheless, one can directly check if a non-empty update exists: Check each new edge individually, since adding more edges cannot remove existing cycles. However, even if a multi-default network can be updated with some edges, it is a hard optimization problem. We define the problem of updating multi-default networks as finding the maximum number of edges that can be included in an update at once:

Problem 1: Let $U_{D,\forall} = (V, E^{old}_{D,\forall}, E^{new}_{D,\forall})$ be a multi-default network update. Find a set $E^{max}_{D,\forall} \subseteq E^{new}_{D,\forall}$, s.t. i) $U^{max}_{D,\forall} = (V, E^{old}_{D,\forall}, E^{max}_{D,\forall})$ is a loop free multi-default network update ii) for all loop free multi-default network updates $U^{other}_{D,\forall} = (V, E^{old}_{D,\forall}, E^{other}_{D,\forall})$ with $E^{other}_{D,\forall} \subseteq E^{new}_{D,\forall}$ it holds that they do not contain more edges, i.e., $|E^{other}_{D,\forall}| \leq |E^{max}_{D,\forall}|$.

Theorem 1: Problem 1 is NP-hard.

Proof: Our proof is a reduction from the classic NP-complete satisfiability problem 3-SAT, in the variant with exactly three pairwise different variables per clause [30]:

1) Consider the routes for destination Y in the triangle-gadget from Figure 3. If node $\overline{X}_i$ updates, then node X_i cannot update without inducing a loop for Y, and vice versa. Choosing one of the two update rules corresponds to a variable assignment for a variable x_i in the instance I of 3-SAT: x_i is either *true* or *false*, but not both.
2) Let C be a clause in the instance I of 3-SAT. If there is a variable assignment S that satisfies I, then updating the triangle-gadgets for the variables according to S does not induce a loop for any destination C in the cycle-gadget for the corresponding clause in Figure 4. If no such variable assignment S exists, then at least one triangle-gadget cannot be updated at all without causing a loop for a destination representing a clause.

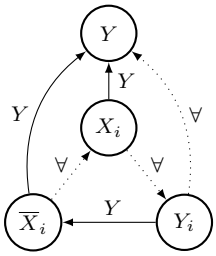


Fig. 3. Triangle-gadget for a variable x_i. New edges are drawn dotted, old solid.

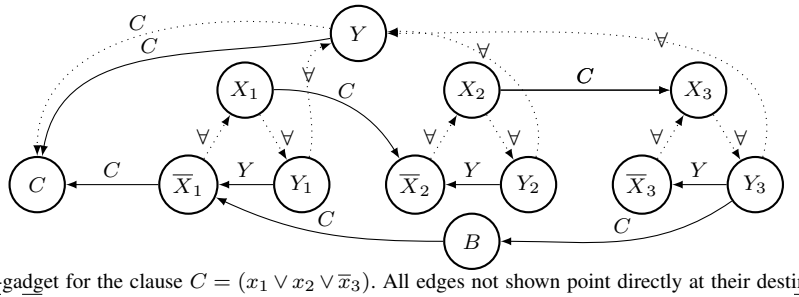


Fig. 4. Cycle-gadget for the clause $C = (x_1 \vee x_2 \vee \overline{x}_3)$. All edges not shown point directly at their destination. Only if all three nodes $\overline{X}_1, \overline{X}_2, X_3$ update their forwarding rule for C, then there is a loop for the label C (via $B - \overline{X}_1 - X_1 - \overline{X}_2 - X_2 - X_3 - Y_3 - B$). E.g., $C = (\overline{x}_1 \vee \overline{x}_2 \vee x_3)$ could only induce a cycle via $B - X_1 - Y_1 - X_2 - Y_2 - \overline{X}_3 - X_3 - B$.

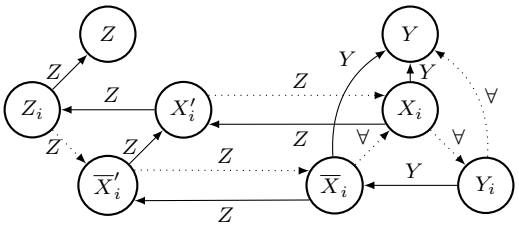


Fig. 5. Extension of the triangle-gadget for a variable x_i from Figure 3. New edges are drawn dotted, old solid. Edges not shown point at their destination. The four possible cycles for destination Z are *i)* X_i, X'_i, *ii)* $\overline{X}_i, \overline{X}'_i$, *iii)* $\overline{X}'_i, X'_i, Z'_i$, *iv)* $\overline{X}'_i, \overline{X}_i, X_i, X'_i, Z_i$. No other new cycles are introduced.

# in sequence	conflicting clauses	variable false	variable true
1	Y, Z, Y_i	$Y, Z, Y_i, \overline{X}_i$	Y, Z, Y_i, X_i
2	$X_i, \overline{X}_i$	$\overline{X}'_i, X_i$	$X'_i, \overline{X}_i$
3	$X'_i, \overline{X}'_i$	X'_i, Z_i	$\overline{X}'_i, Z_i$
4	Z_i	$\emptyset$	$\emptyset$

Fig. 6. Table depicting the fastest possible migration scenarios for the nodes in Figure 5. *i)* X_i cannot update before X'_i, *ii)* $\overline{X}_i$ not before $\overline{X}'_i$, *iii)* Z'_i not before $\overline{X}'_i$ or X'_i, and *iv)* X_i or X'_i must update before $\overline{X}'_i$ and $\overline{X}_i$ and Z_i can all three be updated. Note that Y, Z, Y_i can always update right away. However, if there are conflicting clauses (i.e., the corresponding instance is not satisfiable), then neither $\overline{X}_i$ nor X_i can update right away, but must wait for the next update to be sent out – after the conflicts with the clauses have been cleared, thus requiring a sequence of length four. Else, one could update with a sequence of length three, as shown in the two rightmost columns.

3) Let k be the number of variables in I. If k rules from the nodes $X_i, \overline{X}_i$ in the triangle-gadgets can be updated loop free, then there exists a variable assignment S that satisfies the instance I of 3-SAT. If less than k rules can be updated from the nodes $X_i, \overline{X}_i$ in the triangle-gadgets, then I cannot be satisfied. ■

We now examine interval routing updates: Since the forwarding rules have to be disjoint, we may only apply updates that result in a valid state for each node. I.e., after applying an update, the forwarding rules have to cover all destinations and be disjoint. Removing all current rules and replacing them with a default rule matches this requirement. In a similar fashion, we specify longest-prefix matching updates: A new prefix rule may contain a set of rules it overrides when the rule is inserted at a node. Else, applying an "update" might not change the routing behavior of a node at all.

Corollary 1: Maximizing loop free updates for interval routing or longest-prefix matching is NP-hard.

A. Future Hardware

Even though asynchronicity is inherent in current hardware solutions (e.g., node failures [4] or highly deviating update times [12]), one could imagine these issues being tackled in future work. For example, the method of updating routing information could be decoupled from the remaining computational load of a node, resulting in roughly the same update time for all nodes in a network. Then one would want to find a shortest sequence of precomputed updates that migrate the network from the current old to the desired new routing rules. I.e., the controller will send out a first loop free multi-default update and wait until all affected edge changes are confirmed. This sending out of updates is iterated until all nodes switched their edges to the new desired routing rules. Nonetheless, this problem of updating a network remains hard, i.e., how long is the sequence of updates that are sent out:

Problem 2: Let $U_{D,\forall} = (V, E^{old}_{D,\forall}, E^{new}_{D,\forall})$ be a multi-default network update. Find a sequence of r loop free multi-default network updates $U^1_{D,\forall} = (V, E^{old}_{D,\forall}, E^{new_1}_{D,\forall}), U^2_{D,\forall}, \ldots, U^r_{D,\forall}$ with vertex sets V and corresponding pairwise disjoint new edge sets $E^{new_1}_{D,\forall}, E^{new_2}_{D,\forall}, \ldots, E^{new_r}_{D,\forall}$ s.t. $E^{new_1}_{D,\forall} \cup E^{new_2}_{D,\forall} \cup \cdots \cup E^{new_r}_{D,\forall} = E^{new}_{D,\forall}$ s.t. $r \in \mathbb{N}$ is minimal.

Theorem 2: Problem 2 is NP-hard.

Proof: Note that the construction for the proof of Theorem 1 is not enough to show that Problem 2 is NP-hard: While it is NP-hard to decide if k rules from the nodes $X_i, \overline{X}_i$ in the triangle-gadgets can be updated, the whole network in the proof can always be updated in a sequence of just two updates. In the first step, one would update all nodes (except for the nodes $X_i, \overline{X}_i$ in the triangle-gadgets). Then, in the second step, all the nodes $X_i, \overline{X}_i$ in the triangle-gadgets can be updated, since the possibility of loops in the gadgets created from variables and clauses have vanished after the first update. However, we can extend our construction s.t. for a solution of sequence-length three, all k triangle-gadgets need to update either $X_i, \overline{X}_i$ in the first element of the sequence of updates. Else, a sequence of length four would be needed. The construction is described in the Figures 5 and 6. ■

Corollary 2: It is NP-hard to approximate the length of the sequence of updates needed for Problem 2 with an approximation ratio strictly better than $4/3$.

V. Algorithms for Loop Free Routing Updates

We first consider variants for single-destination updates and then extend the discussion to the other models. While dynamic updates (i.e., update as much as you can at once) are desirable due to fault-tolerance (see Section I, e.g., a node might be

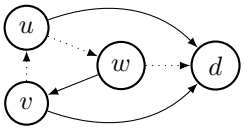


Fig. 7. Illustrating multiple maximal solutions. The nodes u and v cannot update together.

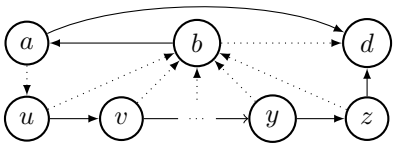


Fig. 8. An update of the nodes a and b is a maximal update, but an update of the nodes $u, v, \ldots, y, z$ and b would be a maximum update.

temporarily unable to update), we also study how to apply all updates in this section. Some proofs are in the Appendix.

We start with single-destination updates: Given an update $U_d = (V, E_d^{old}, E_d^{new})$, find a loop free update $U_d = (V, E_d^{old}, E_d^{'})$ with $E_d^{'} \subseteq E_d^{new}$. We begin by setting $E_d^{'} = \emptyset$:

An update is *maximal*, if adding more edges from E_d^{new} to $E_d^{'}$ violates loop freedom. Maximal updates do not have to be unique, see Figure 7. Node w may switch to the new rule immediately, but not nodes u and v. If they both switch immediately, and w is still using the old rule, we get a loop. So, one of them must wait for w to switch. Either one is fine, i.e. either u must wait for w (and v, w may switch immediately), *or* v must wait for w (and u, w may switch immediately).

Algorithm 1:

1) *Check for an edge $(u,v)_d = e \in E_d^{new}$ if the update $U_d = (V, E_d^{old}, E_d^{'} \cup \{e\})$ is loop free. This loop test can be performed, e.g., by a DFS from node v to find node u on edges with label d.*
2) *If adding e does not introduce a loop, set $E_d^{'} = E_d^{'} \cup \{e\}$.*
3) *Repeat step 1 until all edges were checked.*

Lemma 1: The update calculated by Algorithm 1 is loop free and maximal.

While a maximal solution might seem like a good approach at first glance, it can be far from optimal regarding the number of updates sent out in one flush, see Figure 8: Even for just one destination, a maximum update can be of size $|E_d^{new}| - 1$, but a maximal might just be 2 edges. Can we do better? Since we want to include as many edges as possible, we are essentially solving restricted instances of the NP-complete Feedback Arc Set Problem (FASP) [30]: Given a directed graph, what is the minimum number of edges that needs to be removed to break all cycles. FASP can also be considered in a variant with weighted edges: This allows us to exclude old edges from removal, by giving all old edges an arbitrarily high weight, and all new edges a weight of just 1. The best known approximation algorithm for weighted FAS has an approximation ratio of $O(\log n \log\log n)$ [31], allowing us to enhance the greedy algorithm for maximal updates:

Algorithm 2:

1) *Set the weight of all edges contained in E_d^{old} to ∞, and the weight of all other edges to just* 1.
2) *Calculate a FAS F for the weighted graph $(V, E_d^{old} \cup E_d^{new})$ according to [31].*
3) *Set $E_d^{'} = E_d^{new} \setminus F$.*
4) *Apply Algorithm 1 to make the update maximal.*

Lemma 2: The update calculated by Algorithm 2 is loop free and maximal. The number of removed edges from E_d^{new} can at most be reduced by a factor of $O(\log n \log\log n)$.

Proof: The removal of a FAS implies by definition loop freedom for the network. However, old edges are not allowed to be removed: But since all edges contained in the set of old edges $E_d^{old} = E_d^{old} \cup \left(E_d^{old} \cap E_d^{new}\right)$ have their weight set to infinity, there is always an infinitely better solution than removing any old edge. One would just set the edges being in $E_d^{'}$ to $\emptyset$, which results in a loop free network by definition.

Maximality is ensured by applying Algorithm 1 afterward, which also preserves the loop free property for the network, see Lemma 1. Since Algorithm 1 can only add more edges to the update, and not remove any, the approximation ratio of $O(\log n \log\log n)$ from [31] is still valid. ■

Let us now consider how to apply the whole desired update for a single destination via sending out multiple smaller loop free updates. In the worst case, we will need $|E_d^{new}|$ loop free updates, for example when reversing the links in a ring – only one edge can be updated loop free at a time.

Algorithm 3:

1) *Use Algorithm 1/2 to send out a first update E_{d,g_1}.*
2) *Once a set of nodes has reported back to the central controller that they have performed the rule updates $E'_{d,g_1} \subseteq E_{d,g_1}$ for destination d (and discarded their old rules $E_{d,g_1}^{old'}$), the controller can calculate a current set of old rules. Take into account that the nodes applying the rules $E_{d,g_1} \setminus E'_{d,g_1}$ are still in a limbo state: Either they applied the update already or not, but it is not known due to the asynchronicity until they report in.*
3) *Calculate and send out the next set of update edges $E_{d,g_2} \subseteq \left(E_d^{new} \setminus E_{d,g_1}\right)$ with Algorithm 1/2, which are derived from $\left(V, \left(E_d^{old} \setminus E_{d,g_1}^{old'}\right) \cup E_{d,g_1}, E_d^{new} \setminus E_{d,g_1}\right)$.*
4) *Iterate the process until all new edges are sent out.*

Algorithm 3 computes a series of loop free updates E_{d,g_1}, $\ldots$,E_{d,g_k}, with $\bigcup_{i=1}^{k} E_{d,g_i} = E_d^{new}$. For Algorithms 1 and 2, this can be understood as a dynamic *dependency forest*, which is minimal in the sense that an edge $e \in E_{d,g_j}$ cannot be added to E_{d,g_i}, if $i < j$.

Lemma 3: Iterating either Algorithm 1 or 2 to construct a dynamic dependency forest needs at most $|E_d^{new}|$ non-empty updates to switch the network to the new rules in E_d^{new}.

Proof: If an update is non-empty, then it contains at least one new edge. Thus, $|E_d^{new}|$ non-empty updates suffice to update the network to only new rules. We now show that we can always include at least one new edge in an update, once all sent out rules are applied. Assume that there is no node that is currently applying a new rule, i.e., all nodes that received a new rule for d applied it and reported back to the controller. Thus, no node is in a *limbo* state, where the node was ordered to apply a new rule, but has not successfully reported back yet. For contradiction, let us now assume that Algorithm 1 does not find any new edge to be sent out as an update. Thus, all not yet applied edges were checked, and each would induce a loop when adding it to the network in an update.

However, at least one edge exists that would not induce a loop. For ease of notation, let us call nodes that still need to apply a new rule *old*, and *new* elsewise. Note that currently no

nodes are in limbo. Start from an arbitrary old node, and move along the set of new rules towards the destination d. Since the destination is (by definition) new, along this new-rules path, there must be a last pair of nodes c, p, where the new edge of c points at p, and c is old and p is new. The edge $(c, p)_d$ cannot induce a loop: It points only to nodes which are in the new state already, that is, there are no more old rules which can cause loops. Therefore, Algorithm 1 would have found at least one more edge to be included in a non-empty update to be sent out (and thus, Algorithm 2 as well). ■

Lemma 4: The structure of the dynamic dependency forest is minimal: Any $e \in E_{d,g_j}$ cannot be added to E_{d,g_i}, if $i < j$.

Proof: W.l.o.g. let $e \in E_{d,g_j}$ and consider any update E_{d,g_i} with $i < j$. The set of edges for E_{d,g_i} was maximal, i.e., no more edges could have been added, see the Lemmas 1 and 2. ■

Note that the Algorithms 1, 2, and 3 can be applied to multi-destination network updates by treating them as a set of single-destination network updates: We can compute the variants separately for each label and apply updates in parallel, as edges with different labels will not interfere with each other regarding loop freedom.

A more complex case is where individual rules control routing to multiple destinations and different rules control overlapping sets of destinations. (For non-overlapping destination sets, the situation is similar to above; replace destination sets with a virtual destination.) This situation can emerge in interval-based routing and longest-prefix matching. One can still use adapted versions of Algorithm 1 within Algorithm 3 for maximal loop free updates, but those updates might be empty: In this case, no (loop free) dependency forests to apply all new rules may exist (cf. the network in Figure 2).

We note that in practice, one should divide the multi-graphs $G = (V, E^{old} \cup E^{new})$ into strongly connected components (SCCs), e.g., by implementing Tarjan's algorithm [32]: Edges from different SCCs cannot be part of the same loop, allowing to partition the problem into smaller instances. However, this does not lead to better theoretical approximation bounds.

Also, if we were able to calculate the set of all loops for each label in the multi-graph G induced by an update $G = (V, E^{old} \cup E^{new})$, then we can even improve the approximation ratio for some cases: First, consider each loop for each label as a set of edges, but only add new edges to the sets. The set of old edges was loop free, meaning there are no empty sets. Second, solve the Minimum Hitting Set Problem (MHSP) [30] by choosing a minimum set of new update edges s.t. each loop is broken. MHSP is NP-complete as well, but a greedy approach yields an approximation ratio of $H(|E^{new}|)$ (with some improvement possible [33]), where $H(n)$ is the n^{th} harmonic number, $H(n) \approx \ln n$, cf. [34].

VI. Consistency Space

We now take a broader view of the range of consistency properties. Table 9 helps frame this view. Its rows correspond to consistency properties. We defined loop freedom in Section III; the others are:

	None	Self	Downstream subset	Downstream all	Global
Eventual consistency	Always guaranteed				
Blackhole freedom	Impossible	Add before remove			
Loop freedom (Section V)	Impossible (Lemma 5)		Rule dep. forest		
Packet coherence	Impossible (Lemma 6)			Per-flow ver. numbers	Global ver. numbers [8]
Congestion freedom	Impossible (Lemma 7)				Staged partial moves [3], [11], [12], [13]

TABLE 9
Basic consistency properties & their dependencies.

- **Eventual consistency** No consistency is provided during updates. If the new set of rules computed by the controller are consistent (by any definition), the network will be eventually consistent.
- **Blackhole freedom** No packet should be blackholed during updates. Blackholes occur if a packet arrives at a switch when there is no matching rule to handle it.
- **Packet coherence** The set of rules seen by a packet should not be a mix of old and new rules; they should be either all old or all new rules.
- **Congestion freedom** The amount of traffic arriving at a link should not exceed its capacity. Physical link capacity is a natural limit, but other limits may be interesting as well (e.g., margin for burstiness). Congestion freedom must be maintained without dropping traffic; otherwise, we can trivially meet any limit.

The consistency properties are listed in rough order of strength, and satisfying a property lower on the list often (but not always) satisfies a property above it. Obviously, packet coherence implies blackhole and loop freedom (assuming that the old and new rules sets are free of blackholes and loops). Perhaps less obviously, congestion freedom implies loop freedom because flows in a loop will likely surpass any bandwidth limit. Note that flows may be splittable [35].

However, these properties cannot be totally ordered. Packet coherence and congestion freedom are orthogonal, as packet coherence does not address congestion, and congestion freedom can be achieved with solutions beyond packet coherence. Blackhole freedom and loop freedom are also orthogonal. In fact, trivial solutions for one violates the other—dropping packets before they enter a loop guarantees loop freedom, and just sending packets back to the sender provides blackhole freedom but creates loops.

The columns in Table 9 denote dependency structures. They capture rules at which other switches must be updated before a new rule at a switch can be used safely. Thus, the dependency is at rule level, not switch level; dependencies are often circular at switch level—a rule on switch u depends on a rule on v, which in turn depends on u for other rules. The structures in Table 9 are:

- **None** The rule does not depend on any other update.
- **Self** The rule depends on updates at the same switch.
- **Downstream subset** The rule depends on updates at a subset of switches downstream for impacted packets.
- **Downstream all** The rule depends on updates at all

switches downstream for impacted packets.

- **Global** The rule depends on updates even at potentially all switches, including those that are not on the path for packets that use the rule.

These dependency structures are qualitative, not quantitative. For instance, they do not capture the time it might take for the update to complete. They also assume that switch resources, such as forwarding table memory or internal queues for unfinished updates, are not a bottleneck. Resource limitations induce additional dependencies on the order in which updates can be applied (see below).

In general, update procedures with fewer dependencies (i.e., to the left) are preferable. The cells in Table 9 denote whether a procedure exists to update the network with the corresponding consistency property and dependency structure. We can prove that certain combinations are impossible (proofs are in the Appendix). For example, packet coherence cannot be achieved in a way that rules depend on updates at only a subset of downstream switches.

As we can see, weaker consistency properties (towards the top of Table 9) need weaker dependency structures (towards the left). At one extreme, eventual consistency (i.e., no consistency during updates) has no dependencies at all. Slightly stronger properties, such as blackhole freedom, have dependencies on other rules at the switch itself. A simple procedure for blackhole freedom is to add the new rule in the switch before the old rule is removed. When installed with higher priority, the new rules become immediately usable, without wait.

At the other extreme, maintaining congestion freedom requires global coordination. The intuition here is that maintaining congestion freedom at a link requires coordinating all flows that use it, and some of these flows share links with other flows, and so on.

Interestingly, all cells to the immediate right of impossible cells are occupied in Table 9, which implies that, across past work and this paper, (qualitatively) optimal algorithms for maintain all these consistency properties are known. However, one must not infer from this observation that finding consistent update procedures is a "solved problem," for three reasons. First, some networks may need different properties, for which effective procedures or even best-case structures are unknown (e.g., load balancing across links and maintaining packet ordering within a flow).

Second, even for the properties in Table 9, the picture looks rosy partly because it assumes plentiful switch resources (e.g., forwarding table memory). If switch resources are constrained, maintaining consistency becomes harder. For instance, maintaining blackhole freedom with plentiful switch memory is straightforward and induces no dependencies across switches—we can just add all new rules with high priority before deleting any old rules. But in the presence of switch memory limits, this becomes challenging because introducing a new rule at a switch might require removing another rule first, which can only be removed after having added a new rule at some other switch.

In fact, we can show that in the presence of memory limits, even maintaining a simple property like blackhole freedom is NP-hard. Formally:

Problem 3: Let $c_i \in \mathbb{N}$ be the total interval rule memory of a switch v_i, the combined number of interval rules in current use and the interval rules it can receive in one update. Let $G = (V, E)$ be the directed graph on which packets can be routed, with the destinations $D \subseteq V$ and the sources $S \subseteq V$ for the packets. In one round, a central controller can send out a set of any interval rules as an update to each node in the network. What is the minimum number of rounds, to migrate the network from a set of blackhole free old rules to a new set of blockhole free rules, if no blackholes should be introduced during migration and routing should be possible at all times?

Theorem 3: Problem 3 is NP-hard.

Proof: The proof for Theorem 3 is based on a reduction from the NP-hard directed Hamiltonian Cycle problem (HC), cf. [30]: Given a directed graph $G = (V, E)$, is there a cycle that visits each node exactly once? The construction with further details is shown in Figure 10: It is possible to migrate blackhole free in two rounds if and only if there is a Hamiltonian Cycle in G, thus allowing to first use the cycle for intermediate routing via default rules, and then installing the new rules; Else it will take three rounds, one for each new rule. Thus, it is NP-hard to decide whether one can migrate in two or three rounds, even if the diameter is just two. The construction for the memory limit of $c = 4$ for all nodes in V can be directly extended to any $c \in \mathbb{N}$ with $c \geq 4$. ■

Furthermore, note that blackhole freedom is easy to guarantee for each node in the presence of default rules, if one does not care about routing: Just set a default rule to any neighboring node. While packets might not arrive at all (and in addition violate other consistency properties, e.g., congestion freedom), blackhole freedom is guaranteed.

Corollary 3: It is NP-hard to approximate the number of rounds needed for Problem 3 with an approximation ratio

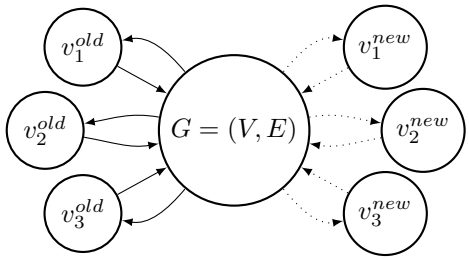


Fig. 10. The center node represents the graph $G = (V, E)$ from an instance I of the directed Hamiltonian Cycle problem, with nodes $v_1, \ldots, v_n$. The sets of edges to (n each) and from ($n/3$ each) the outer six nodes are bundled into single edges in this figure. Each node in $V = S = D$ has a memory limit c of four rules, with S being the set of packet sources and D being the set of packet destinations. The solid edges represent the edges used for the three old rules $\forall v \in V$, the dotted edges the edges used for the three new rules $\forall v \in V$. All nodes in V currently use the three nodes v_1^{old} (for $v_1, \ldots, v_{(n/3)}$), v_2^{old} ($v_{(n/3)+1}, \ldots, v_{(2n/3)}$), v_3^{old} ($v_{(2n/3)+1}, \ldots, v_n$) on the left for 2-hop routing to the respective destinations in $D = V$, and want to migrate to use the nodes v_1^{new} (for $v_1, \ldots, v_{(n/3)}$), v_2^{new} ($v_{(n/3)+1}, \ldots, v_{(2n/3)}$), v_3^{new} ($v_{(2n/3)+1}, \ldots, v_n$) on the right for 2-hop routing.

strictly better than $3/2$.

Third, the table only shows the qualitative part of the story, ignoring quantitative effects, which may be equally important. Even though [8] and [3] both have global dependencies, [8] can always resolve the dependencies in two rounds, whereas [3] may need more stages. Because of these three reasons, we believe that what is presented in this paper is just the tip of the iceberg for consistent updates in Software Defined Networks.

VII. An architecture for SDN updates

We have argued that maintaining consistency during rule updates is a key hurdle towards realizing the promise of SDNs. The question is: how can we accomplish this in a flexible, efficient manner? A straightforward possibility is that a single software module (controller) decides on new rules and then micro manages the update process in a way that maintains consistency. However, this monolithic architecture is undesirable because it mixes three separable concerns — i) the rule set should be policy-compliant; ii) rules updates should maintain the desired consistency property; iii) the update process should be efficient, which depends on the asynchronicity in the network.

We propose an alternative architecture (Figure 11) with three parts, one for each concern above: i) the *rule generator* produces policy-compliant rules; ii) the *update method selector* chooses the method of how to apply the rules, based on data from past updates; and iii) the *update executor* schedules the updates efficiently in a dynamic fashion, taking current asynchronicity into account.

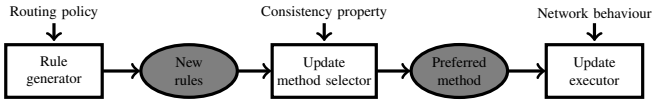


Fig. 11. Proposed dynamic architecture for SDN updates

The update method selector proceeds in two steps. It first generates, using the old rules and collected data from past updates of the network, a model of the current state of the network. This includes, e.g., the mean and variance of applying an update to a switch or the amount of unallocated memory/bandwidth. In the second step, multiple methods of applying the update are checked and simulated on the model of the network. Depending on the outcome, a preferred method for updates is selected: For example, if the current amount of free memory on switches is small, packet stamping is not a viable update method. However, if a long chain of links needs to be reversed loop free, and memory is not an issue, packet stamping might be the best way to proceed. In this step, it is also possible to issue *helper rules*, that are neither in the old or new set of rules, but allow consistent updates via a specific method. Consider the network in Figure 2: One can prevent loops by breaking a single (default) rule into one for each of the other destinations, introducing these rules during the update process and then removing them later.

The update executor computes a maximal set of updates that can be sent out immediately with the selected method, using the old rules, the new rules, and the desired consistency property. Once a set of nodes reported back on the successful implementation of the new rules, another batch of updates can be sent out into the network. Since the update process is a dynamic one, faulty nodes only induce a limited delay, independent parts of the network can still be updated. Nodes that did not report back yet have to be considered in a limbo state: Either they applied the new rules already or not, but to not break consistency properties, one has to assume that they are in both the new and the old state at the same time.

An example for an update executor would be Algorithm 3: Maximal sets of loop free updates are sent out each time nodes report back about the successful implementation of rules, inducing a minimal dependency structure in form of a dynamic dependency forest.

VIII. Evaluation

We took Rocketfuel ISP topologies with intra-domain routing weights [36] and considered link failures in these topologies, with our goal being loop free network updates from pre- to post-failure least-cost routing.

Figure 12 plots the distribution of the length of dependency chains that emerge across ten trials, where a randomly selected link was failed in each. We see that roughly half of the updates depended on 0 or 1 other switch, and 90% of all forwarding rules were dependent on at most 3 other switches. In contrast, had we used Reitblatt's procedure [8], which ensures the stronger property of packet coherence, rules would have had to wait for all other switches (well over a hundred in some cases), and a single slow switch can impede everyone.

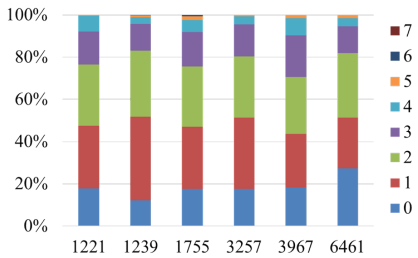


Fig. 12. Chain lengths in loop free updates in six Rocketfuel topologies. The x-axis label denotes the ASN.

Francois et al. [18] evaluated their work on a tier-1 ISP with 200 nodes and 800 links, resulting in chain lengths of 14. We had a chain length of at most 7, even for tier-1 ISPs such as ASN 1239 (Sprintlink) with 547 nodes and 1647 links.

IX. Summary

We argued that consistent updates in Software Defined Networks is an important and rich area for future research. We highlighted the trade-off between the strength of the consistency property and the dependency structure it imposes, and developed minimal algorithms for loop freedom. For the basic consistency properties of loop and blackhole freedom, we showed that fast updates are NP-hard optimization problems. We also sketched an architecture for consistent updates and showed that our loop freedom algorithm performs well in evaluations on ISP topologies.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their helpful comments, which helped us to improve the presentation of this paper. We would also like to thank Stefan Schmid and Stefano Vissicchio for pointing us to [15], [16], [17] shortly before this article was accepted for publication. Klaus-Tycho Förster was supported in part by Microsoft Research.

REFERENCES

[1] M. Borokhovich and S. Schmid, "How (Not) to Shoot in Your Foot with SDN Local Fast Failover," in *OPODIS*, 2013.
[2] M. Casado, N. Foster, and A. Guha, "Abstractions for software-defined networks," *Commun. ACM*, vol. 57, no. 10, pp. 86–95, Sep. 2014.
[3] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *SIGCOMM*, 2013.
[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hoelzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined WAN," in *SIGCOMM*, 2013.
[5] R. Mahajan and R. Wattenhofer, "On consistent updates in software defined networks," in *HotNets*, 2013.
[6] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *USENIX NSDI*, 2005.
[7] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. van der Merwe, "The Case for Separating Routing from Routers," in *SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, 2004.
[8] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *SIGCOMM*, 2012.
[9] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in!" in *10th ACM Workshop on Hot Topics in Networks*, 2011.
[10] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates," in *HotSDN*, 2013.
[11] S. Brandt, K.-T. Förster, and R. Wattenhofer, "Augmenting anycast network flows," in *ICDCN*, 2016.
[12] X. Jin, H. Liu, R. Gandhi, S. Kandula, R. Mahajan, J. Rexford, R. Wattenhofer, and M. Zhang, "Dionysus: Dynamic Scheduling of Network Updates," in *SIGCOMM*, 2014.
[13] S. Brandt, K.-T. Förster, and R. Wattenhofer, "On Consistent Migration of Flows in SDNs," in *INFOCOM*, 2016.
[14] T. Mizrahi, O. Rottenstreich, and Y. Moses, "TimeFlip: Scheduling network updates with timestamp-based TCAM ranges," in *INFOCOM*, 2015.
[15] A. Ludwig, J. Marcinkowski, and S. Schmid, "Scheduling Loop-free Network Updates: It's Good to Relax!" in *PODC*, 2015.
[16] S. Vissicchio and L. Cittadini, "FLIP the (Flow) Table: Fast LIghtweight Policy-preserving SDN Updates," in *INFOCOM*, 2016.
[17] A. Ludwig, S. Dudycz, M. Rost, and S. Schmid, "Transiently Secure Network Updates," in *Sigmetrics*, 2016.
[18] P. François and O. Bonaventure, "Avoiding transient loops during the convergence of link-state routing protocols," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1280–1292, 2007.
[19] P. François, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *Computer Communication Review*, vol. 35, no. 3, pp. 35–44, 2005.
[20] P. Francois and O. Bonaventure, "Loop-free convergence using oFIB," *Internet-Draft, IETF*, 2011.
[21] L. Vanbever, S. Vissicchio, C. Pelsser, P. Francois, and O. Bonaventure, "Lossless Migrations of Link-state IGPs," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1842–1855, Dec. 2012.
[22] N. Santoro and R. Khatib, "Routing without routing tables," SCS-TR-6, Carleton University, Ottawa, Tech. Rep., 1982.
[23] C. Gavoille, "A survey on interval routing," *Theor. Comput. Sci.*, vol. 245, no. 2, pp. 217–253, 2000.
[24] M. Flammini, G. Gambosi, and S. Salomone, "Boolean Routing," in *WDAG*, 1993.
[25] P. Fraigniaud and C. Gavoille, "A characterization of networks supporting linear interval routing," in *PODC*, 1994.
[26] J. Van Leeuwen and R. B. Tan, "Interval routing," *The Computer Journal*, vol. 30, no. 4, pp. 298–307, 1987.
[27] D. Comer, Ed., *Internetworking with TCP/IP - Principles, Protocols, and Architectures, Fourth Edition*. Prentice-Hall, 2000.
[28] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010.
[29] V. Fuller and T. Li, "RFC 4632, Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan," 2006.
[30] M. R. Garey and D. S. Johnson, *A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.
[31] G. Even, J. Naor, B. Schieber, and M. Sudan, "Approximating minimum feedback sets and multicuts in directed graphs," *Algorithmica*, vol. 20, no. 2, pp. 151–174, 1998.
[32] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.
[33] A. Srinivasan, "Improved approximations of packing and covering problems," in *STOC*, 1995.
[34] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.
[35] R. Cohen and G. Nakibly, "Maximizing restorable throughput in mpls networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 568–581, 2010.
[36] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *IMW*, 2002.

X. APPENDIX FOR SECTION 5

Proof of Lemma 1:
We start with loop freedom: The invariant of the algorithm is that the current edges in the network are without loops. The invariant is true at the beginning, since no new edge is included, and the old edges form an in-tree to the destination d. When a new egde $(u, v)_d$ is added, a now existing loop must contain this edge, i.e., there is a path from v to u. If a DFS starting at v cannot reach u, then there is no path from v to u, and the network is loop free. We now look at maximality: The algorithm checks each edge once if it can be added without inducing a loop. Consider an edge $e = (x, y)_d$, that is being tested w.l.o.g. as the i-th edge, but cannot be added to the network, because it would induce a loop $x, y, z, \ldots, x$. If e is being tested again after the $(j-1)$-th edge, with $i < j$, could e be added to a loop free network without inducing a loop in the network? No, because it would still induce the same loop, as edges were never removed, only possibly added. ■

XI. APPENDIX FOR SECTION 6

Lemma 5: Loop freedom depends on other nodes.

Proof: In Figure 1, node x depends on node y. ■

Lemma 6: Packet coherence depends on all non-trivial downstream switches.

Proof: Let u be a switch router that is non-trivial, in the sense that u is affected by a rule change, i.e. u's old rule differs from its new rule. If the source starts to route packets according to the new rule, switch u will forward the packets wrongly, or drop them, which is not packet coherent. ■

Lemma 7: Congestion freedom depends on all switches.

Proof: Let f be a flow that wants to use a new path p, or increase its capacity on an existing path. The network may be able to adapt to flow f, however, only if other flows use different paths as well, which in turn may (recursively) move even other flows (some of which have no single switch/link in common with the new path p). As such, any f may potentially depend on any single switch in the network. ■

Arrange Your Network Updates as You Wish

Shouxi Luo, Hongfang Yu, Long Luo, Lemin Li
Key Laboratory of Optical Fiber Sensing and Communications, Ministry of Education
University of Electronic Science and Technology of China, Chengdu, P. R. China

***Abstract*—Updating network configurations responding to dynamic changes is still a tricky task in SDN. During the update process, in-flight packets might misuse different versions of rules, and "hot" links could be overloaded due to the unplanned update order. As for the problem of misusing rule, recently proposed suggestions like *two-phase mechanism* and *Customizable Consistency Generator (CCG)* have provided *generic* and *customizable* solutions. Yet, there does not exist an approach that is flexible to avoid the transient congestion on hot links respecting to diverse user requirements like guaranteeing update deadline, managing transient throughput loss, etc.; controllers urgently need one.**

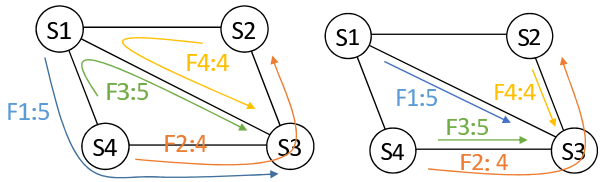
In this paper, we propose CUP, Customizable Update Planner, to seek the solution. Different from prior approaches that adopt fixed designs for a single purpose like optimizing the update speed (e.g., Dionysus) or avoiding congestions (e.g., zUpdate, SWAN), CUP introduces generic linear programming models to formulate user-specified requirements and the update planning problem. By solving these customized models, CUP is able to plan network updates according to a large fraction of user requirements, such as guaranteeing deadlines, prioritizing operation orders, managing throughput loss, etc., while avoiding transient congestion. We prototype CUP on Ryu and employ it to arrange updates for networks built upon Mininet. Results confirm the flexibility of CUP while indicating that it always obtains the "best" update plans following the user's wish.

I. Introduction

Reconfiguring forwarding rules in networks responding to dynamic demands such as periodical traffic optimization, unexpected failover, is always a tricky task for operators [1]–[6]. Recent trends toward Software Defined Networking (SDN) seem to provide a promising solution for network management—with a logical central controller, operators can directly operate the forwarding rules on all switches. Even so, the network is still an asynchronous system in essence. It is difficult to synchronize the changes to flows from different ingress switches. Therefore, when migrating a group of flows to their new paths, even if the network is safe both before and after the reconfiguration, some "hot" links could be overloaded during the update process in case new flows move in before those old ones move out [2]–[4].

As an example, consider the toy case shown in Fig. 1. On executing WAN optimizations [3], the controller wants to update the network's configuration from Fig. 1a to Fig. 1b. For simplicity, we assume that the network uses tunnel-based routing and all necessary tunnels have already been established. If the controller carries out the update in one-shot, link S4-S3 or S1-S3 might be overloaded during the update, corresponding to the case that switch S4 happens to change F3 to its new path before S1 moving F1 away from link S4-S3, or vice versa. The congestion can not be evaded by simply letting F1 and F3 be switched to their new paths at exactly the same time [7]—because the incoming packets of F2 and F3, together with the in-flight packets of F1, could still congest S4-S3 until F1 drains; and so does S1-S3.

ISBN 978-3-901882-83-8 © 2016 IFIP



(a) Current State (b) Target State

Fig. 1: A network update example. Each link has 10 units of capacity and flows are labeled with their sizes. If the controller carries out the update in one-shot, link S1-S3 or S4-S3 will be overloaded during the update.

Such a type of congestion disappears following the completion of update, but its destructibility lasts long—burst traffic leads to serious queuing delay, and even, packet drops, which will let involved TCPs' windows collapse, or worse, kill flows. These bad influences are not desirable, especially for real-time applications. Accordingly, carrying out network reconfigurations without introducing transient congestion is a fundamental function required by SDN controller.

Planning network updates to avoid transient congestion is never an easy task. Recent approaches like zUpdate [2] and SWAN [8] try to solve the problem by introducing a sequence of intermediate configurations, among which, the update from a former stage to the latter must always be congestion-free. To ensure such a stage sequence exists, they require part of the link capacity to be left vacant, which results in a great waste of link capacities [8, 9]. Furthermore, the intermediate configurations they introduce will greatly complicate the update process, and might even disturb user's QoS—e.g., an intermediate path might have a larger latency than both the initial and target ones. In contrast, Dionysus [3] and Atomip [4] address the challenges by scheduling updates in thoughtful orders without bringing in additional stages. For instance, by executing the update illustrated in Fig. 1 following the 3-round sequence of [F4→F1→F3], no link would be overloaded and no extra paths are introduced. Order arrangement provides a more practical solution. However, it is not always the panacea because such a congestion-free operating sequence does not always exist. Indeed, due to the various update scenarios and user demands that a controller would deal with, simply arranging the update operations, or introducing intermediate stages, is far from enough for a practical solution. We argue that, a practical planner should have these properties.

1) Effective to handle deadlock and deadline. First of all, the planner must be able to find feasible congestion-free solutions for any given task. On one hand, in some update scenarios, there does not exist a congestion-free sequence [3, 4]. For instance, in the case of Fig. 1, if the demand of either F1 or F3 increases to 6, it is impossible to migrate the network to its target routing state by arranging the execution order without overloading S1-S3 or S4-S3. This is a deadlock in update planning. On the other hand, even though congestion-free schedules are found, they may not meet the deadline requirements. This is because to remove overloads, the controller can not switch flows belonging to round-$(i+1)$ to their new paths until flows moved out from these paths in round-i have exited. Suppose in-flight packets require about τ units of time to exit from a path on average; then, it would take about $k \cdot \tau$ for the entire network to perform a k-round update, even without considering the time of rule installations. Such an update delay/duration might be unacceptable for time-critical cases like failover routing [10]. Therefore, *on planning updates, the planner should have the ability to break deadlocks and guarantee deadlines.*

Fortunately, for any update, by limiting the rates of some flows at their senders or traffic shapers, controllers can always obtain a congestion-free update sequence that involves fewer rounds and satisfies the deadline requirements. Indeed, there is a trade-off between the time an update takes, and the throughput the network has to drop (induced by congestion or rate-limiting). For example, one can carry out the update request demonstrated in Fig. 1 within 2 rounds by limiting the rate of either F2 or F4 to 0 (e.g., when F2's rate is limited to 0, [F3→F1, F4] is congestion-free), or even perform the update within 1 round by limiting the rate of both F2 and F4 to 0. This example gives us a valuable insight: *the planner should have the ability to trade throughput loss for update speed*.

2) Expressive to deal with user-specified requirements. As infrastructure, today's network is shared by numerous customers while simultaneously carrying various kinds of traffic. To be a universal tool for controller, the update planner should be extensible and easy to adapt to user-specified requirements. As an example, consider the case of removing transient congestion for the update illustrated in Fig. 1 again. Provided the reconfiguration is time-sensitive and required to complete within 1 round, the controller has to reduce some flow rates to avoid congestion. Suppose this is an instance of inter-datacenter traffic optimization [8], in which both F1 and F3 are *interactive* traffic while F2 and F4 are *background* traffic, and the operator prefers to minimize the amount of interactive traffic disturbed by the update. In such a scenario, the planner should temporarily reduce the rates of F2 and F4 to 0 to execute the update, i.e., limit the rates of {F1, F2, F3, F4} to $\{5, 0, 5, 0\}$. On the contrary, if F2 and F4, instead of F1 and F3, are interactive, the result would be $\{1, 4, 1, 4\}$. As another example, if all flows share the same class and a fairness alike policy is expected [11], the planner should set their rates to $\{\frac{5}{14}, \frac{4}{14}, \frac{5}{14}, \frac{4}{14}\}$, with the target of letting the decrease of throughput be fairly shared in proportion.

Indeed, due to network's diversity, such a special constraint of rate-limiting is only the tip of an iceberg. In practice, there are plenty more kinds of user-specified demands (about the update execution time or throughput loss) that a controller would deal with. It follows that, *on planning rate-limiting schemes, the planner should be flexible enough to suit various update scenarios, as well as user-specified demands*.

3) Efficient to scale up. Last but not least, to be practical, the planner must be time-efficient to find feasible solutions for update requests in time. In consideration of that the size of today's network might be really huge (e.g., Datacenter or backbone), the planner needs to easily scale up.

As the first step, this paper proposes CUP, Customizable Update Planner, to help controller deal with various updating requirements. CUP suggests adopting generic methods such as *two-phase mechanism* [5, 6] to enforce rule consistency, and focuses on eliminating the transient congestion during updates. Distinguished from existing solutions proposed for fixed targets, CUP is effective and expressive to deal with deadlock, deadline, prioritization, and many other user-specified requirements as Table I summaries (Note that, proposals focusing on enforcing rule consistency are not listed, e.g., CCG [12].). We analyze various demands and realize that, besides consistency, what users/operators concern about the implementation of an update, no matter how complex it is, generally involves two types of fundamental issues—i) *when a flow could enjoy its new path(s)* and ii) *how its throughput would be impacted during the update process?*

At a high-level, CUP provides an expressive user-friendly language, with which, customers and operators can describe their own requirements easily and explicitly. When the network is to be updated, CUP maps these high-level requirements into the essence (involved) flows, and translates them into low-level linear constrains. At its core, CUP builds a couple of generic linear programming models to formulate the update request while capturing constrains from users. Via solving these customized models, CUP obtains a congestion-free update execution plan that explicitly follows the user's wish.

Roughly, CUP's model involves two parts, *Order Scheduler* and *Rate Manager*, which respectively answer the two basic problems mentioned above. On planning an update, *Order Scheduler* first determines the operation order respecting to time-related requirements. If congestion-free sequences are found, *Order Scheduler* outputs the one involving the minimum rounds; otherwise, it chooses the sequence causing least overload on links. For the overloaded traffic, *Rate Manager* then figures out the optimal rate-limiting scheme that is able to erase the congestion while satisfying all throughput-related requirements. As the core of both *Order Scheduler* and *Rate Manager* is to solve a single Linear Program (LP), with high performance LP solvers, CUP obtains solutions within polynomial time and is able to scale up.

We prototype CUP upon Ryu[1] and use it to plan updates for networks conducted by Mininet [13]. Results show that CUP is quite flexible to exactly meet user-specified requirements, while effective to outperform existing approaches.

In summary, we make three contributions in this paper.

[1] An open-source SDN controller framework, https://osrg.github.io/ryu/

TABLE I: Summary of previous approaches and comparison to CUP.

#Proposal	Introduce intermediate status?	Effectiveness		Expressiveness
		Handle deadlock	Deal with deadline	Meet user-specified requirements
zUpdate [2]	Yes	No	No	No
SWAN [8]	Yes	Yes	Single deadline for all	No
GI [9]	Yes	Yes	Single deadline for all	No
Dionysus [3]	No	Yes	No	No
ATOMIP [4]	No	Yes	Single deadline for all	No
CUP	**No**	**Yes**	**Per-flow deadline**	**Yes (any time- and rate- related requirements)**

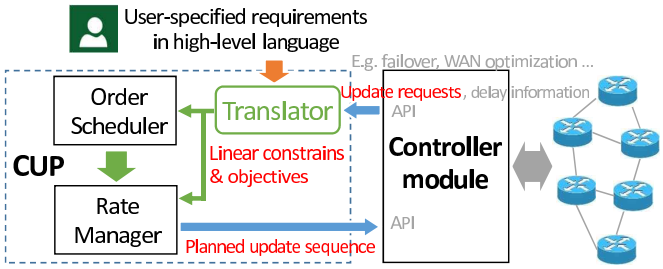


Fig. 2: The workflow of CUP on planning updates.

Grammar

pol	::=	$(s_1; \ldots; s_n)$	CUP Policy
s	::=	$t \mid r$	Rule
t	::=	$T(m) \leq val \mid T(m_1) \leq T(m_2)$	Time Related Req.
r	::=	$R(m) \geq amap \mid R(m_1) \geq R(m_2)$	Rate Related Req.

Notation

m	:	a match string/predicate specifying flows
val	:	a value specifying a deadline requirement
$T(m)$	:	the waiting time before m enjoys the new path(s)
$amap$	:	the keyword specifying objectives (as much as possible)
$R(m)$	:	the rate-limit setting of flow(s) defined by m

Fig. 3: Syntax of CUP high-level language.

- **Abstraction:** We show how to express various user-specified updating requirements with a high-level language, and show how to dynamically translate them into low-level linear constraints (Section II).
- **Model:** We propose generic linear programming models to formulate and solve the customized update planning problem, with which, controllers obtain the "best" update plan explicitly following user's wish (Section III).
- **Evaluation:** We show that our CUP tool is flexible and effective to make update plans for "real" networks built by Mininet (Section IV).

II. FLEXIBLE CUP

In CUP, network users as well as operators describe their desired properties about the update with the high-level CUP language; they can change the clauses at any time. On planning a network update, at the first step, CUP "compiles" the user's codes to figure out their exact "meaning" in this instance. After that, CUP employs back-end solvers, *Order Scheduler* and *Rate Manager*, to find the update processing plan that exactly follows the user's wish. Roughly, the entire workflow of how CUP produces is as Fig. 2 shows.

In the following, we present the high-level language in Section II-A and show the compilation process in Section II-B. After that, in next section, we introduce how CUP solves the planning problems and discuss how it handles multi-tenants and concurrent update requests.

A. High-level language

CUP language (Fig. 3) provides end-users and operators with an easy way to specify their requirements on configuring the network. A CUP policy is a collection of rules, in which, each term specifies a specific requirement, of either the activation time of new paths or the degradation of throughputs, for a group of packets. CUP uses a regular expression on the match fields of packet header to define the involved packets. For instance, $*$ defines all packets pass through the network; dstTCP=80 defines all web access traffic; srcIP=10.0.0.1/24 $\wedge$ dstIP=20.0.0.11 defines those packets from network 10.0.0.1/24 to destination 20.0.0.11; and srcIP=10.0.0.2 $\vee$ dstIP=10.0.0.4 defines the traffic from 10.0.0.2, or to 10.0.0.4.

For the update of a collection of packets specified by m, there are two basic types of indicators that customers and operators might concern: 1) how long it would wait before enjoying the new path(s), and 2) how its throughput (i.e., rate) would be limited to avoid transient congestion. CUP uses $T(m)$ and $R(m)$ to denote, respectively. Using their relation expressions, these two basic elements can generate other complicated requirements. For instance, $T(m_1) \leq T(m_2)$ says, flows matched with m_1 should be switched into the new paths "no later than" those matched with m_2, while $T(m_2) \leq val$ indicates the waiting time before m_2 switched should be "no larger than" val. Similarly, $R(m_1) \geq R(m_2)$ implies the effective bandwidth of m_1 during the update should "no less than" that of m_2, while $R(m_3) \geq amap$ means the user would like the effective bandwidth of m_3 be maximized.

CUP language is simple yet expressive for most requirements. As examples, revisit the toy update cases of Fig. 1. With CUP language, users can formulate their own requirements precisely and concisely as the instances in Table II illustrate.

B. Dynamic translator

High-level CUP policies are compiled into low-level restrictions, which tell the planner how to process each flow's reconfiguration is in line with user requirements. To achieve this, the most challenging task is to figure out the exact time cost of migrating a flow. CUP employs the approach of pre-installing new rules then triggering two-phase reconfigurations to address the problem. In this part, we first present how to make the estimation of reconfiguration's time cost possible in

TABLE II: Examples of CUP language on describing update cases shown in Fig. 1.

#	Update scenarios	Policy expression
1	Minimize transient congestion without deadline requirements on the update process.	$(R(*) \geq amap)$
2	Let interactive flows,F_2 and F_4, enjoy new paths no later than 1 unit time, while minimizing the impacts on their throughputs (e.g., inter-DC WAN optimization [8, 14]).	$(T(m_{F2} \vee m_{F4}) \leq 1;$ $R(m_{F2} \vee m_{F4}) \geq amap)$
3	Execute all flow migrations no later than 1 unit time, and let the throughput loss be shared in proportion since they are in the same class.	$(T(*) \leq 1;$ $R(m_{F1}) \geq amap;$ $R(m_{F2}) \geq amap;$ $R(m_{F3}) \geq amap;$ $R(m_{F4}) \geq amap)$

Section II-B1, then introduce the way of binding high-level requirements with flows and translating them into low-level linear constraints in Section II-B2.

1) Estimating time cost of traffic migration: As Section I and Fig. 1 have shown, to not overload any link during the update, the controller has to wait the flow that moved out from a link exits, before moving other flows in. Thus, the time cost of migrating a flow to its new path(s) mainly involves two parts of i) waiting the moved-out traffic exits (if any); and then ii) installing rules to shift the flow to its new path(s).

As for the first part of draining time, we can simply use the well-known One-Way Delay (OWD) as an approximation, which can be estimated at end hosts [15, 16], or at edge switches in OpenFlow-enabled networks. CUP suggests adopting *two-phase update mechanism* to guarantee strong rule consistency (refer to Appendix A in [17] for the discussion). On carrying out an N-rounds flow migration, at the first step, CUP pre-installs the new configurations and sets rate-limits. Supposing the time of installing/modifying a rule from the controller is ϵ, the total time cost of this step is ϵ because all rule installations (for both new paths and rate-limits) can perform in parallel. Thus, the rest operations for each round are to i) wait a draining time then ii) touch some flows' ingress switches to activate their new paths. Provided the largest OWD in network is τ, we get the point that flows migrated in the kth round would enjoy their new paths at time $k \times \tau + (k+1) \times \epsilon$. Consequently, if a flow's deadline requirement on the update process is val, we know that the controller should make sure it get migrated no later than round $\lfloor \frac{val-\epsilon}{\tau+\epsilon} \rfloor$.

In practice, the time cost of modifying a rule on physical switches is usually inconstant [3, 14, 18, 19]. Yet, recent studies have shown its long-tailed characteristic [3]. That is to say, simply choosing the 95th percentile value (or other thresholds) as the estimated time is reasonable in most cases. Moreover, since OpenFlow-style control is still in its early stages, most switch software and SDKs are not optimized for dynamic table programming yet [14]. Some effects have been put on improving this and we argue that future switches will be more stable and fast for table changes [18, 20].

As yet, we have found a way to estimate the time cost of migrating a flow based the network's maximum OWD and ingress's rule modification delay. In real networks, both types of delays can be measured by the controller. With this information, CUP is able to translate the absolute deadline requirements into round requirements. For simplicity, hereafter, all deadline requirements we discuss in this paper are in the form of round number.

TABLE III: The key notations of the network model

Notation	Description
M_R^{due}	the set of predicates (m) holding $T(m) \leq val$
M_R^{amap}	the set of predicates (m) holding $R(m) \geq amap$
MP_T	the set of $\langle m_x, m_y \rangle$ pairs holding $T(m_x) \leq T(m_y)$
MP_R	the set of $\langle m_x, m_y \rangle$ pairs holding $R(m_x) \geq R(m_y)$
$\hat{N}_m^{due}$	the round deadline for flow matching with m
$f \in F$	the set of all current flows in the network
$F(m)$	the set of all flows matching with predicate m
t_f	the demand of flow f
r_f	the rate-limit setting of f during the update
r_m^*	the rate-limit setting for all flows matching with m
$e \in E$	the set of all (directed) links in the network
c_e	the capacity of link e
$t_{f,e}$	the load of f on link e before the update
$t'_{f,e}$	the load of f on link e after the update
F^B	the set of flows that will not be updated/migrated
F^U	the set of flows that will be updated/migrated
$F^U(m)$	the set of to-be-updated flows matching with m
FP_T	$\forall \langle f_i, f_i \rangle \in FP_T$: f_i should be updated no later than f_j
N_f^{due}	f's update deadline, in the form of round number
$y_{f,k}$	whether f has been updated in round-k
$t_{f,e,k}$	the (maximum) load of f on e in round-k

2) Mapping requirements to each flow: Now, we show how CUP maps user requirements into each flow. The basic notations that CUP's model uses are summarized in Table III.

Lexical analysis and preprocessing. CUP first parsers user-specified policies to get the semantics. Obviously, there are four types of constraints on the flow predicates, indicating the absolute update deadline (i.e., $T(m) \leq val$), the relative update order in "no-later-than" form (i.e., $T(m_x) \leq T(m_y)$), relative rate-limiting setting in "no-less-than" form (i.e., $R(m_x) \geq R(m_y)$), and the expected targets that should be optimized (e.g., $R(m) \geq amap$). Without loss of generality, we let M_T^{due} be the set of predicates holding the relation of $T(m) \leq val$, and M_R^{amap} be the set of predicates holding $R(m) \geq amap$. As well, we further use MP_T and MP_R to denote the set of predicate pairs (e.g., $\langle m_x, m_y \rangle$) that have the relation of $T(m_x) \leq T(m_y)$ and $R(m_x) \geq R(m_y)$, respectively. As discussed above, for a deadline requirement on flows specified by predicate m, CUP can transfer it into a round number requirement with Equation (1), where $\hat{\tau}$ is the network's measured maximum OWD and $\hat{\epsilon}$ is the measured 95th rule modification delay.

$$\hat{N}_m^{due} = \lfloor \frac{val_m - \hat{\epsilon}}{\hat{\tau} + \hat{\epsilon}} \rfloor \quad (1)$$

Basic network model. We assume that the network, G, is hosting a set of flows F with links E. The rate of flow $f \in F$ is denoted by t_f while the capacity of link $e \in E$ is denoted by c_e. By letting $t_{f,e}$ be the traffic load of flow f on link e, the network's state can be formulated as $S = \{t_{f,e} | \forall (f \in F, e \in E)\}$. Then, a network update is to change its state from S to $S' = \{t'_{f,e} | \forall (f \in F, e \in E)\}$ by rerouting some flows, or changing their traffic split ratios in the

case of multi-path routing. For the update of $S \mapsto S'$, let F^U be the set of updated flows and F^B be the set of unmodified flows. Obviously, there must be $F^U \cap F^B = \emptyset$, $F^U \cup F^B = F$, and $t_{f,e} = t'_{f,e}$ for $\forall (f \in F^U, e \in E)$. We assume that the update of flow f is required to be finished within N_f^{due} rounds, and use bin variable $y_{f,k} (1 \leq k \leq N_f^{due})$ to indicate whether f ($f \in F^U$) has been migrated/updated in the k-th round. By defining $y_{f,0} = 0$ for convenience, we get the constraints as Equation (2) and (3) show.

$$\forall k, f \in F^U : y_{f,k} \in \{0, 1\} \tag{2}$$

$$\forall f \in F^U : 0 = y_{f,0} \leq y_{f,1} \leq \ldots \leq y_{f,N_f^{due}} = 1 \tag{3}$$

Besides, we let r_f denote the proportion of rate-limiting that flow f would be set to during the update. Then, after rate-limiting is enabled, the total load of f would be reduced to $t_f \cdot r_f$, and the subpart on e before and after the update would also decrease to $t_{f,e} \cdot r_f$ and $t'_{f,e} \cdot r_f$, respectively.

$$\forall f : 0 \leq r_f \leq 1 \tag{4}$$

Embedding user-specified requirements. In networks, flows are also defined by predicate strings of the packet header fields. By checking whether a flow's predicate intersects with the user-specified predicate, CUP figures out which flows are involved with that rule. For rule predicate string m, we denote $F(m)$ as the set of flows that it intersects with, and $F^U(m)$ as the subpart of to-be-updated flows in $F(m)$. Then, via Equation (5), CUP gets the set of rules that a flow is matched with and gets the exact deadline requirement of each flow. It should be noted that, the entire update process will never exceed $|F^U|$, the number of flow to be updated. So, in case the estimated round calculated from user policies is larger than $|F^U|$, or no deadline is required, N_f^{due} will be set to $|F^U|$.

$$N_f^{due} = \min(|F^U|, \min_{\forall m \in M_T^{due} : f \in F^U(m)} \hat{N}_m^{due}) \tag{5}$$

As for the "no-later-than" order requirements, $T(m_x) \leq T(m_y)$, if two to-be-updated flows, f_i and f_j, happen to hold the relations of $f_i \in F^U(m_x)$ and $f_j \in F^U(m_y)$, it means they have order-dependency on the update active time, namely, $y_{f_i,k} \geq y_{f_j,k}$ for all feasible k. Let FP_T be the set of such order-dependent flow pairs; CUP can easily get it by calculating Equation (6). Then, all "no-later-than" requirements are as Equation (7) shows.

$$\begin{aligned} FP_T = \{\langle f_i, f_j \rangle | \exists \langle m_x, m_y \rangle \in MP_T; \\ f_i \in F^U(m_x); f_j \in F^U(m_y)\} \end{aligned} \tag{6}$$

$$\forall (f_i, f_j) \in FP_T, k \leq \min(N_{f_i}^{due}, N_{f_j}^{due}) : y_{f_i,k} \geq y_{f_j,k} \tag{7}$$

Now, CUP deals with rate/throughput related requirements. Same to the case of time-related predicates, the predicate m in a rate-specified rule also might match with multiple flows at the same time. We denote the collection of involved flows as $F(m)$ and regard them as a "virtual" aggregated flow. For this "virtual" flow, we further use r_m^* to present what its rate-limit would be during the update process. Then the two types of throughput requirements could be formulated as Equation (8) and (9) show, in which r_m^* is defined by Equation (10) and $amap$ is the index/variable that should be optimized.

$$\forall (m_i, m_j) \in MP_R : r_{m_i}^* \geq r_{m_j}^* \tag{8}$$

$$\forall m_i \in M_R^{amap} : r_{m_i}^* \geq amap \tag{9}$$

$$r_m^* = \frac{\sum_{\forall f \in F(m)} r_f \cdot t_f}{\sum_{\forall f \in F(m)} t_f} \tag{10}$$

So far, CUP has translated all user-specified requirements into low-level flow-based constraints, which are all linear.

III. Efficient Solver

To handle various updates, CUP needs a generic yet efficient solver. However, the design is not easy since planning updates is computationally intractable in ordinary sense—even answering the question of whether there exists a congestion-free solution for a given update is NP-hard as Theorem 1 says.

Theorem 1. *Determining whether there is a congestion-free update order scheduling that meets user-specified deadline is NP-Hard in ordinary sense.*

Proof. Refer to Appendix B in [17] for details. □

Corresponding to the fact that planning an update involves two parts of 1) finding an execution order and 2) computing the relevant rate-limiting scheme, CUP heuristically decouples the original problem into two parts as Fig. 2 shows. On planning a group of flow migrations, the *Order Scheduler* module first determines which round each flow should be moved in, based on user-specified time-related requirements. If there exists congestion-free sequences, *Order Scheduler* outputs the one with the minimum rounds; otherwise, it suggests the sequence causing smallest traffic overloads. Then, for the congested traffic, *Rate Manager* further finds the optimal rate-limiting scheme that makes the update free of congestion, respecting to throughput/rate-related rules.

A. Order Scheduler

The first step of planning update to prevent transmit congestions is to evaluate what link loads would be during the update procedure. For flow $f \in F^U$, we let $t_{f,e,k}$ indicate its maximum possible load on link e when preforming the reconfiguration of round k. Then, the maximum (possible) load on link e in this round is $\sum_{f \in F^B} t_{f,e} + \sum_{f \in F^U} t_{f,e,k}$.

$$t_{f,e,k} = \begin{cases} t_{f,e} - y_{f,k-1} \cdot (\max(t_{f,e}, t'_{f,e}) - t'_{f,e}) \\ \quad + y_{f,k} \cdot (\max(t_{f,e}, t'_{f,e}) - t_{f,e}) & Changed\ ind. \\ t_{f,e} - y_{f,k-1} \cdot t_{f,e} + y_{f,k} \cdot t'_{f,e} & Otherwise \end{cases} \tag{11}$$

The calculation of $t_{f,e,k}$ for round k has two formulations depending on f's update senses as Equation (11) shows. In both formulations, it is certainly that f's load on link e equals $t_{f,e}$ if f has not been migrated yet, i.e., $y_{f,k-1} = y_{f,k} = 0$, or equals $t'_{f,e}$ if its migration has completed, i.e., $y_{f,k-1} = y_{f,k} = 1$. The difference exists in the case when f happens to be migrated in round k, i.e., $y_{f,k-1} = 0$ and $y_{f,k} = 1$, and the link is used by both f's old path(s) and new path(s).

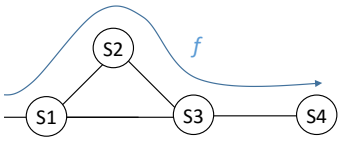


Fig. 4: An example of that the updated flow is not *changed independently*: move f from path S1-S2-S3-S4 to S1-S3-S4.

In datacenter networks, the multiple paths between two end-hosts usually share the same hops and packets traveling through them are likely to experience the similar delay [2]. Accordingly, the load of f on link e during the update is either $t_{f,e}$ or $t'_{f,e}$. In this condition, f is *changed independently* [2] on link e, and its maximum possible load during the update is $\max(t_{f,e}, t'_{f,e})$, corresponding the upper case of Equation (11). However, the situation of WAN is quite different, in which multiple paths of a source-destination pair generally have distinct delays. In the worst case, the load of flow f on e would reach $t_{f,e} + t'_{f,e}$. As an example, consider the case of rerouting flow f from path S1-S2-S3-S4 to S1-S3-S4 shown in Fig. 4. On switching f to its new path, because of the transmission and buffer delays, incoming packets traveling through S1-S3, together with the in-flight packets on sub-path S1-S2-S3, would contribute a total load of $t_{f,e} + t'_{f,e}$ on link S3-S4. Fortunately, by comparing the new network configuration with the old one, CUP knows whether a flow is *changed independently* or not. Then, the right expression of $t_{f,e,k}$ for flows and links can be decided.

On computing the update order, CUP tries to minimize the overloaded traffic on links while optimizing the total required rounds. Provided o_e is the amount of overloaded traffic on link e (whose capacity is c_e), there are many alternative formulations that capture the link load situation of the entire network—E.g., $\sum_{\forall e} o_e$, $\max_{\forall e} o_e$, $\sum_{\forall e} \frac{o_e}{c_e}$, and $\max_{\forall e} \frac{o_e}{c_e}$. CUP adopts $\max_{\forall e} o_e$. With this design, even if the network is failed to apply the rate-limiting schemes, the scheduled update order will still let the transient congestions be distributed on most links, so that the overloaded packets are more likely to be held by switch buffers.

$$\forall e, k > 0 : \sum_{f \in F^B} t_{f,e} + \sum_{f \in F^U} t_{f,e,k} \leq c_e + o_e; o_e \geq 0 \quad (12)$$

Obviously, this order scheduling problem is naturally to be formulated as a MIP (Mixed Integer linear Program) as Fig. 5 shows, where γ is a small factor ($0 \leq \gamma \ll 1$) and the tail of $-\gamma \cdot \sum_{\forall (f,k)} y_{f,k}$ is to let flows be migrated as soon as possible. For the schedule of a small scale update, we can obtain the optimal order by directly solving this MIP with efficient solvers. However, as finding the optimization scheduling order is theoretically NP-hard, the computation process becomes quite time-consuming when the network scales up. To find scheduling orders quickly, we relax the original MIP into a Linear Program (LP), and develop an efficient heuristic solution based on the relaxed LP's outputs. Due to the lack of space, the detail of heuristic algorithm follows in our technical report (Appendix C) [17].

In practice, a simple way to achieve both efficiency and effectiveness on order scheduling is to employ a "dual-core" trick. For each planning request, CUP can perform the MIP solving and heuristic computation, simultaneously. If MIP completes within a certain time (e.g., one second), CUP gets the optimal results; otherwise, CUP chooses the heuristic result and stops the task of MIP solving.

$$\begin{cases} \textbf{Input:} & F^B, F^U, FP_T, \{c_e\}, \{t_{f,e}\}, \{t'_{f,e}\}, \{N_f^{due}\} \\ \textbf{Output:} & \{y_{f,k} | \forall (f \in F^U, k)\} \\ \text{Minimize} & \max_{\forall e} o_e - \gamma \cdot \sum_{\forall (f,k)} y_{f,k} \\ \text{Subject to} & (2), (3), (7), (11), \text{and } (12), \end{cases}$$

Fig. 5: Schedule update orders to minimize the link overloads. γ is a small constant: $0 < \gamma \ll 1$.

$$\begin{cases} \textbf{Input:} & F^B, F^U, \{c_e\}, \{t_{f,e}\}, \{t_{f,e,k}\}, MP_R, M_R^{amap} \\ \textbf{Output:} & \{r_f | \forall f\} \\ \text{Maximize} & amap + \varrho \times \min_{\forall f} r_f \\ \text{Subject to} & (4), (8), (9), (10), \text{and } (13). \end{cases}$$

Fig. 6: Manage transient congestions in each update round $\{r_f | \forall f\}$ explicitly following user's requirement. ϱ is a small constant: $0 < \varrho \ll 1$.

B. Rate Manager

Once the update order is determined, CUP gets the value of $\{t_{f,e,k} | \forall (f, e, k)\}$. The next issue is to find a rate-limiting scheme avoiding congestion respecting to user's requirements. As defined in Section II-B2, r_f is the ratio that flow f should decrease to for removing transient congestions; then, the straightforward solution to obtain the optimal rate-limiting scheme for user-specified requirements is to solve the corresponding LP shown in Fig. 6.

$$\forall e, k > 0 : \sum_{f \in F^B} r_f \cdot t_{f,e} + \sum_{f \in F^U} r_f \cdot t_{f,e,k} \leq c_e \quad (13)$$

Note that, when no $amap$-based rule is specified, CUP adopts $R(*) \geq amap$ by default, which results in minimizing the total throughput loss. In some cases, there might be multiple rate-setting schemes that obtain the same optimal $amap$. CUP adds a tail of $\varrho \times \min_{\forall f} r_f$ (ϱ is a small positive constant) into the objective to gain the one let flows share the loss of throughput in proportion.

About efficiency. So far, we have built a generic solver made up of *Order Scheduler* and *Rate Manager* for CUP. Obviously, the core of both *Order Scheduler* and *Rate Manager* is solving LPs, which can be efficiently done within polynomial time by leveraging fast solvers like CPLEX and MOSEK. Consequently, the entire solver is a polynomial time approach as well. Furthermore, there are several simple yet efficacious tricks that CUP can employ to simplify the model and accelerate the computation. For example, if a link would never be overloaded during the update, CUP can exclude its related constraints from the model safely. We call such links *non-critical*, and they can be determined by Equation (14) easily. Corresponding, if a flow only encounters with *non-critical* links, it is also *non-critical* and there is no need to limit its rate. So, CUP can remove its constraints from the rate-manage model. As well, if a to-be-updated flow is *non-critical* and does not have "no-later-than" relation with others, it can be migrated directly in

the first round without planning computations.

$$E_{non\text{-}crit.} = \{\forall e \mid \sum_{\forall f \in F^B} t_{f,e} + \sum_{\forall f \in F^U} \max_{\forall k} t_{f,e,k} \leq c_e\} \quad (14)$$

Multi-tenant. In practice, a network might be shared by multiple tenants (or virtual operators) simultaneously [21]. The requirements specified by a tenant should only impact its own updates and own traffic. In such cases, CUP would look into the tenant information when embedding policies. As for CUP's solver, *Order Scheduler* is able to handle this directly because there is no difference on the sub-problem of order scheduling; however, *Rate Manager* needs a modification as the rate management problem is a *multi-objective optimization problem* now—$\max\ (amap_1, amap_2, \ldots, amap_n)$. Multiple-objective optimization has been studied for very long time and there are so many solutions, such as *scalarization*, *no-preference methods*, *priori methods*, *etc* [22]. In this paper, CUP simply adopts the approach of linearly *scalarizing* [22] the multiple objectives into the single objective of $\max \sum_{\forall i} w_i \cdot amap_i$, where $w_i \geq 0$ stands for the weight of the ith tenant. By simply pursuing this scalarizated objective, CUP supports multi-tenant updates. We note that there is room to improve and CUP is flexible to be upgraded.

Concurrent updates. In general, a "fat" update request involving many flow migrations would be planned to execute in more than one round. As the network configuration is volatile, new update request is likely to occur before the current "fat" one completes. This should be handled appropriately and immediately as some new flow migration requests might have urgent deadline requirements. CUP adopts the generic two-phase mechanism [5] to implement the reconfiguration of each round, which naturally supports update streams. Accordingly, CUP can immediately deal with a new request by just regarding it together with these unperformed rounds as a fresh request; rule consistency is always guaranteed.

IV. Evaluation

In this section, we implement CUP based on Ryu, and conduct virtual networks with Mininet to test CUP. Our results indicate that CUP is flexible enough to handle user-specified time- and throughput- requirements. Moreover, CUP is very effective. On each type of requirement, CUP always significantly outperforms the variant of Dionysus which is modified to handle that requirement type.

A. Implementation

We prototype CUP upon Ryu 3.26, and employ it to plan traffic migrations for toy virtual networks on Mininet 2.2 [13].

Network setup. When switches start up, the controller installs default routes and tunnel rules via OpenFlow 1.3. We let end-hosts send UDP packets with each other in steady rates to simulate the case of backbone traffic in WAN, and use VLAN tags to implement tunnel-based forwarding for them. We assume that the network adopts multi-path routing, in which ingress switches split and assign a flow to its sub-tunnels respecting to tunnel weights. Then, updating a flow is only to reconfigure its tunnel weights at the ingress, so that each update is consistent in essence [5, 6].

To carry out weighted traffic splitting on Open vSwitch, the controller installs a group of exact-match rules specifying the tunnel for each microflow.[2] Unfortunately, this approach makes rule management on ingress complex as the update of a single flow might trigger the modification of a collection of microflow rules. We address the problem by using the Multiple Flow Table mechanism provided by OpenFlow switches (supports start from OpenFlow 1.1). Basically, rules in an ingress switch are either stored in Table 0 or Table 1 depending on their types. In normal, forwarding functional rules like tunnels and default routes reside in Table 1, and these microflow rules that realize traffic splits and tunnel selections, together with a lower priority all-$*$ whose action is "goto Table 1", reside in Table 0. When a flow's splitting weights are to be updated, the controller first installs microflow rules that implement the new weights in Table 1, then installs a high-priority wildcard rule with action "goto Table 1" into the first table to "guide" involved packets to the new weights. After that, the controller silently modifies the actions of those unmatched microflow rules in Table 0 following the new weights, then deletes the previously installed wildcard rule and microflow rules. Following this, we make rules easy to manage and guarantee the consistency property during weight reconfigurations.

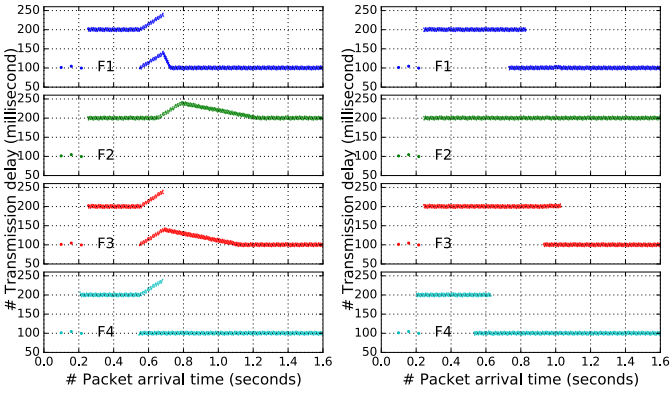
Benchmark schemes. We implement CUP's algorithm in Python and employ Mosek as the backend solver for LPs. As a benchmark, we implement the schedule algorithm of Dionysus. Although it is designed for dynamic scheduling of updates, under the situation that new rules are pre-installed and ingress switches share the similar time cost on enabling new configurations for flow, Dionysus would also derive a round schedule together with a rate limiting scheme for each update in advance [3]. If the obtained round number is larger than the deadline requirement, we assume that Dionysus adopts its deadlock-break mechanism for help—limit the rates of flows whose scheduled time would miss the deadline to zeros, and perform all their migrations in the last round.

B. Case study

To evaluate how transient congestion caused by unplanned updates would influence the traffic, we first conduct experiments for the toy update cases shown in Fig. 1. Note that all virtual hosts and switches in Mininet use the shared CPU and bandwidth resources for simulation [13]. To avoid resource competition between them and to highlight the results, we set link bandwidth to 5 Mbps with 100 ms delay, and let port buffer size be large enough to hold all overloaded traffic. Accordingly, in the case of no congestion, the transmission delay of all old paths is about 200 ms, same to the network's maximum OWD, and that of the new paths is about 100 ms.

Fig. 7a shows the transmission delay of packets in each flow when the controller sends the "activate the new path" commands for {F1, F3, F4} in *One Shot* at the 0.4 s.

[2] In tests, the traffic from a host to another is equally dispersed over 20 UDP flows, and its ingress switch holds a corresponding number of microflow rules for traffic splitting. Thus, the accuracy of traffic-splitting is 0.05.



(a) Unplanned One-Shot update (b) Planed update $(R(*) \geq amap)$

Fig. 7: Transient congestion during unplanned updates.

About 150 ms later, receivers get packets through the new paths. Obviously, the latency of packets in all flows increase during the update process. That is to say, they all entered queues because of transient congestion. In the test, we set no artificial control delay between the controller and switches (however, there is still a delay about 50 ms for each flow table modification from CUP sending the command via REST API) so that all flows enjoy their new paths almost at the same time. As a result, the newly incoming packets of F1 together with the in-flight packets of F3 and F4 overload Link S1-S3, while F1's in-flight packets together with the newly incoming packets of F2 and F3 overload Link S4-S3. In practice, the activation time of new rule might be distinct on switches; transient congestion happens once a flow moves in the hot link before the old in-flight packets exists. And these overloaded packets in high speed network can be really huge, which would quickly eat up switch buffers and result in heavy packet loss [3].

As a comparison, Fig. 7b shows the case of migrating flows in order of [F4→F1→F3], which is the result planned by both Dionysus and CUP under the policy of $(R(*) \geq amap)$. In this case, the controller triggers flow migrations round by round, and waits the maximum OWD time (200 ms) between them. Following the plan, the update process takes about 600 ms to complete, but avoids all transient congestion.

Then, we look into the case of planning updates with time- and throughput- requirements. Provided the update request appear at the 0.4 s, and the operator wants all flows to enjoy their new paths no later than 300 ms; that is to say, all flow migrations must be carried out within one round,[3] and rate-limits are needed to avoid congestion. Fig. 8 and Fig. 9 show the results planned by CUP under user-specified policies $(T(*) \leq 1; R(*) \geq amap)$ and $(T(*) \leq 1; R(m_{F2} \vee m_{F4}) \geq amap)$, respectively. In the case of Fig. 8, all flows share the same importance and the operator prefers the total throughput be reduced as less as possible. With the objective function shown in Fig. 6, CUP's *Rate Manager* lets the throughput loss be shared by all flows in proportion as Fig. 8b shows, where $\frac{\Delta y}{\Delta x}$ stands for the flow rates observed by the sender or receiver—about $\{\frac{5}{14}, \frac{4}{14}, \frac{5}{14}, \frac{4}{14}\}$. Different from Fig. 8, Fig. 9 demonstrates the case that F1 and F4 are background traffic while F2 and F4 are interactive whose throughput should be keep as much as possible. As the results show, CUP finds the update plan exactly following the operator's wish. In contrast, Dionysus will handle the requirements in a rough way—completely kill F1 and F2 to avoid congestion.

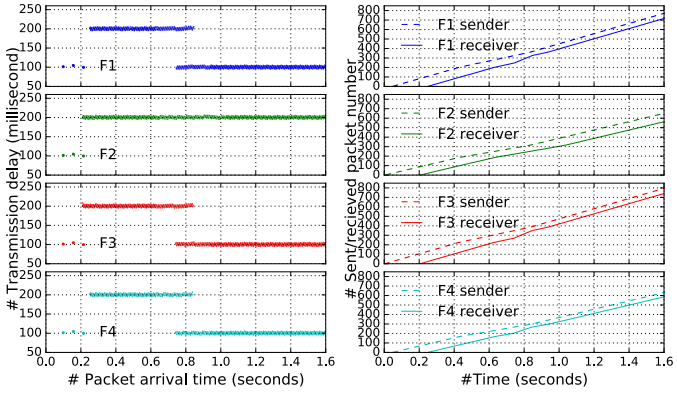
[3]It takes about 200 ms to pre-install new rules and wait rate-limits coming into force; then less than 100 ms is left for performing the updates.

C. CUP flexibility

To investigate the flexibility of CUP, we further employ it to plan updates for a small WAN [3, 9], which involves 8 nodes and 14 links as Fig. 10 illustrates. In this case, each link is assumed to have the capacity of 10 Mbps and delay of 200 ms. We consider the case of WAN optimization, where ingress switches split the traffic to a destination among its 4-shortest paths to pursue load balancing. Because of lacking real traffic matrices, we assume that all the possible paths of a source-destination share the equal weight initially, and use *gravity model* [4] to synthesize the current traffic demands, which make the maximum link load be 99% in the old configuration. Then, the update scenario is to reconfigure traffic split weights to the new one that reduces the maximum link load to the minimized value, 78%. The longest path(s) in tests involves 4 links; accordingly, the network's maximum OWD is 800 ms. For each link e, we consider it as *unchanged independently* for flow f, if f has more than one path going through e and these paths hold distinct lengths (i.e., delays).

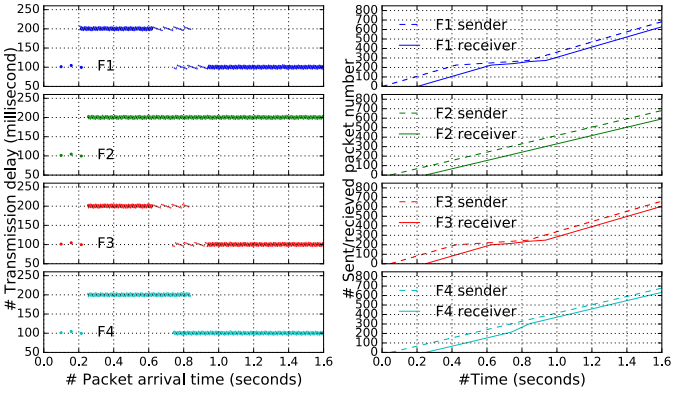
When no update deadline is required, CUP finds a congestion-free plan involving 5 rounds without limiting flow rates, while Dionysus obtains a 6-round plan that achieves the same goal. Then, we artificially add deadline requirements to all flows and compute the propitiation of network throughput that CUP, as well as Dionysus, has to abandon for congestion freedom. Numerical results indicate that CUP outperforms Dionysus about $3\times$ on reducing the impact of network throughput as Fig. 11 shows. CUP is excellent because its *Rate Manager* always obtains the optimal rate-limiting scheme respecting to user's requirements. On the contrary, Dionysus just randomly kills some flows to move on. In addition, Dionysus would never touch the rate of the un-updated flows. But in some cases, slowing down some of them really helps.

We also study the cases that some traffic is background and the operator wishes interactive traffic be less impacted during the update. To this end, we assume that a certain percentage of traffic between each source-destination pair is background, then calculate how many round CUP, as well as Dionysus, would need to perform congestion-free reconfiguration without reducing the throughput of interactive traffic. Fig. 12 demonstrates the results. It implies that, with the proportion of background traffic increasing, the round number required by CUP rapidly decreases. And after the background traffic accounts for half of the traffic, CUP always performs congestion-free updates in one round without reducing the rates of interactive flows. In contrast, Dionysus can not achieve this because of its unawareness of user-specified requirements. If we pre-limit the rates of background traffic to zeros, Dionysus then obtains small update rounds as CUP does. However, similar to the cases shown in Fig. 11, such a solution is far from good because too many flows are killed unnecessarily.



(a) Impact on transmission delay (b) Impact on flow rate

Fig. 8: Plan under policy: $(T(*) \leq 1; R(*) \geq amap)$, i.e., all migrations should be finished within 1 round and the total network throughput should be as-much-as-possible.



(a) Impact on transmission delay (b) Impact on flow rate

Fig. 9: Plan under $(T(*) \leq 1; R(m_{F2} \vee m_{F4}) \geq amap)$, i.e., all migrations should be finished within 1 round and the total throughput of F2 and F4 should be as-much-as-possible.

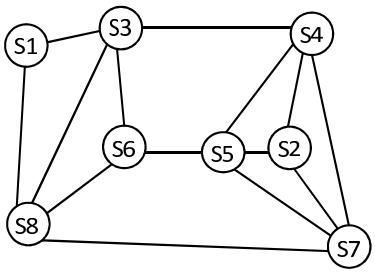


Fig. 10: WAN topology in [3].

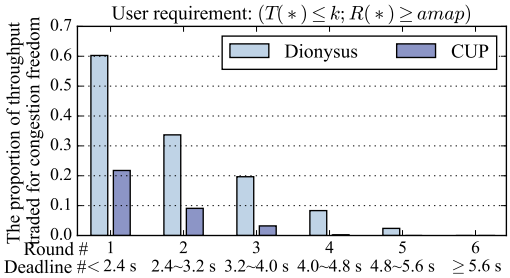


Fig. 11: Throughput loss V.S. update speed.

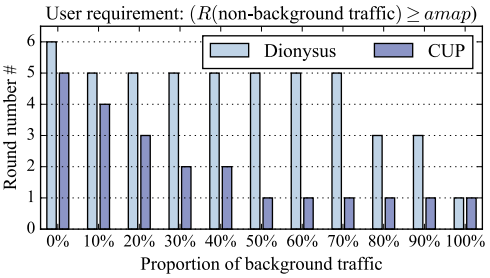


Fig. 12: Impact of background traffic.

V. Conclusion

As transient congestions are prone to occur during SDN updates, controllers are in urgent need of a planner to handle the trouble. We argue that planning the reconfiguration process respecting to specified requirements is an import issue. In this paper, we have analyzed the desired properties of such planners and proposed a case design–CUP. CUP translates high-level user-specific requirements into linear constraints and formulates the planning problem as generic linear programs. By solving customized LPs, CUP is flexible to obtain "best" plans for a large fraction of updates.

Acknowledgements. This work was supported in part by the 973 Program (2013CB329103), 863 Program (2015AA015702, 2015AA016102), NSFC (61271171, 61271165, 61571098), Ministry of Education - China Mobile Research Fund (MCM20130131), China Postdoctoral Science Foundation (2015M570778), Fundamental Research Funds for the Central Universities (2682015CX072), Science and Technology Program of Sichuan Province (2016GZ0138).

References

[1] S. Raza, Y. Zhu, and C.-N. Chuah, "Graceful Network State Migrations," *IEEE/ACM Trans. Netw*, vol. 19, no. 4, pp. 1097–1110, Aug 2011.
[2] H. H. Liu *et al.*, "zUpdate: Updating data center networks with zero loss," in *SIGCOMM*, Aug 2013, pp. 411–422.
[3] X. Jin *el al.*, "Dynamic scheduling of network updates," in *SIGCOMM*, Aug 2014, pp. 539–550.
[4] L. Luo, H. Yu, S. Luo, and M. Zhang, "Fast lossless traffic migration for SDN updates," in *IEEE ICC*, June 2015, pp. 5803–5808.
[5] S. Luo, H. Yu, and L. Li, "Consistency is not easy: How to use two-phase update for wildcard rules?" *IEEE Communications Letters*, vol. 19, no. 3, pp. 347–350, March 2015.
[6] M. Reitblatt *el al.*, "Abstractions for network update," in *SIGCOMM*, Aug 2012, pp. 323–334.
[7] T. Mizrahi, E. Saat, and Y. Moses, "Timed consistent network updates," in *Proc. ACM SOSR*, 2015, pp. 21:1–21:14.
[8] C.-Y. Hong *el al.*, "Achieving high utilization with software-driven WAN," in *SIGCOMM*, Aug 2013, pp. 15–26.
[9] J. Zheng, H. Xu, G. Chen, and H. Dai, "Minimizing transient congestion during network update in data centers," in *Proc. 23rd ICNP*, Nov 2015.
[10] H. H. Liu *el al.*, "Traffic engineering with forward fault correction," in *SIGCOMM*, Aug 2014, pp. 527–538.
[11] V. T. Lam *el al.*, "Netshare and stochastic netshare: Predictable bandwidth allocation for data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 5–11, Jun. 2012.
[12] W. Zhou *el al.*, "Enforcing customizable consistency properties in software-defined networks," in *NSDI*, May 2015, pp. 73–85.
[13] N. Handigol *el al.*, "Reproducible network experiments using container-based emulation," in *CoNEXT*, 2012, pp. 253–264.
[14] S. Jain *el al.*, "B4: Experience with a globally-deployed software defined wan," in *SIGCOMM*, Aug 2013, pp. 3–14.
[15] O. Gurewitz, I. Cidon, and M. Sidi, "One-way delay estimation using network-wide measurements," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2710–2724, June 2006.
[16] A. Pathak *el al.*, "A measurement study of internet delay asymmetry," in *Proc. 9th PAM*, 2008, pp. 182–191.
[17] S. Luo *et al.*, "Arrange your network updates as you wish," http://shouxi.name/publication/cup-tr.pdf, Tech. Rep., Dec 2015.
[18] J. H. Han *el al.*, "Blueswitch: enabling provably consistent configuration of network switches," in *Proc. ACM/IEEE ANCS*, May 2015, pp. 17–27.
[19] M. Kuzniar *et al.*, "What you need to know about SDN control and data planes," Tech. Rep., 2014, EPFL-REPORT-199497.
[20] R. Bifulco and A. Matsiuk, "Towards scalable SDN switches: Enabling faster flow table entries installation," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 343–344, Aug. 2015.
[21] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" in *OSDI*, 2010, pp. 1–14.
[22] Wikipedia, "Multi-objective optimization," https://en.wikipedia.org/wiki/Multi-objective_optimization, 2015, [Online; accessed 23-Nov-2015].

Action Computation for Compositional Software-Defined Networking

Heng Pan*†, Gaogang Xie*, Peng He*, Zhenyu Li*, Laurent Mathy‡
*ICT, CAS, China, †University of CAS, China, ‡University of Liége, Belgium
{panheng, xie, hepeng, zyli}@ict.ac.cn, laurent.mathy@ulg.ac.be

***Abstract*—Software-defined networking (SDN) envisions the support of multiple applications collaboratively operating on the same traffic. Policies of applications are therefore required to being composed into a rule list that represents the union of application intents. In this context, ensuring the correctness and efficiency of composition for match fields as well as the associated actions is the fundamental requirement. Prior work however focuses only on the composition of match fields and assumes simple concatenation for action composition. We show in this paper that simple concatenation can result in incorrect behavior (for parallel composition) and inefficiency (for sequential composition) for actions composition. To address this issue, we formalized the action composition problem and prove a feasibility condition on the composition of rule actions. We then propose a graph-based approach that facilitates fast composition of action lists without action redundancy. Our proposed approach has been integrated into the CoVisor code base and the evaluation results show its fitness for purpose.**

***Index Terms*—Software-defined Networking, composition, action**

I. Introduction

Software-Defined Networking (SDN) decouples control logic from the forwarding devices to simplify network management and enable complex network applications [1]. Such a separation allows the control plane software and data plane hardware to evolve quickly and independently. Recent interest in SDN has moved to the implementation of various SDN applications upon controllers written in different programming languages. The vision of SDN is to construct an SDN "App Store" [2], [3], [4] for network management services. Similar to the Android Market or the Apple Store, network administrators could download applications suited to their needs from the SDN "App Store" and deploy them into the network. For example, a single network could run simultaneously a firewall written in Java on OpenDaylight [5], a routing application written in Python on Ryu [6] or a monitoring application in C on NOX [7].

To realize this vision, a mechanism that compiles different processing logics of applications to cooperate correctly in the data plane is essential. In general, there are two types of approaches towards such a mechanism: *top-down* and *bottom-up*. The top-down approaches use either domain specific programming languages [8], [9], [10] or a specific programming framework [11], [12], to express each application as a program (module) or an expressive equivalent (*e.g.* graph in [11]). These programs are then translated into a set of low-level OpenFlow rules representing the union of the `intents` of the applications. The bottom-up approaches on the other hand utilize SDN hypervisors [13], [14], lying between the controllers and the underlying forwarding devices, to compose policies[1] into a prioritized list of (OpenFlow [15]) rules. Nonetheless, both types of approaches essentially face the same challenge: composing multiple policies, each representing the intent of an application (program, module), into a single rule list that represents the union of these intents.

In the context of composing multiple policies, two types of composition operators have been proposed in existing SDN programming frameworks: parallel (+) and serial (>>) [9], [16], [10], [17], [18]. Parallel composition gives the illusion that each member policy acts on its own separate copy of the traffic while sequential composition enables multiple policies to operate on traffic in sequence. For example, if the hypervisor applies a composition configuration as follows: `Firewall >> (Monitoring + Routing)`, packets will be processed first by Firewall, and then operated on by Monitoring and Routing concurrently.

A policy consists of match fields and the associated atomic actions, which enable programmers to design abundant expressive behaviour represented as a sophisticated action list. A practical composition mechanism should ensure that the composed rule of multiple policies is correct (in terms of application intents) and efficient (in terms of packet processing) for both match fields and action lists. Prior work on policy composition [13], [19], [14], [11] however mostly discusses how to merge the match fields of rules from different member polices and how to calculate the priority of the composed rules, leaving action composition much overlooked. Indeed, the action composition essentially boils down to the "union" of actions (often implemented as the concatenation of actions) in the previous work. This observation was corroborated by inspection of the released code of the CoVisor system [20] and many language implementations such as Frenetic [21].

We show that simple concatenation for composing action lists not only cannot preserve the semantics (or interests) expected by the original member policies but also can result in wasted compute cycles in the resource constrained forwarding path environment of switches. For example, consider one member policy rule's action list is {`push_vlan(1),`

[1]To simplify our discussion, we use the terms "policy" and "application" interchangeably.

ISBN 978-3-901882-83-8

`tcpdst ← 80, fwd(1)`} while the other is {`dstip ← 10.0.0.1, tcpdst ← 80, fwd(2)`}. If the corresponding two rules are composed to operate on packets in parallel, and the actions lists are simply concatenated, the result becomes {`push_vlan(1), tcpdst ← 80, fwd(1), dstip ← 10.0.0.1, tcpdst← 80, fwd(2)`}, which obviously violates the semantics of the second original rule, since the packet appearing on port 2 are different from the one that would have been generated by this second rule, had it been operating alone. This is because the second rule forwards the input packets with modified IP destination address `10.0.0.1` to port 2, while the composed action list forwards the input packets with both the appropriately modified IP destination address *and* an added vlan header to port 2. Obviously, the second action of `tcpdst ← 80` is redundant, which wastes the compute cycles of underlying switches. Overall, to the best of our knowledge, there is no mechanism to effectively compute action sequences for composing SDN policies.

Motivated by our observations, we in this paper address the challenge of correct and efficient action composition in the context of policy composition. our contributions are four-fold:

1) We show and prove, feasibility conditions on the composition of rule actions in SDN networks. By extension, this result also applies to the feasibility analysis of the composition of the policies themselves;
2) We derive a feasibility test, which can be applied to the "on-the-fly" composition of rules.
3) We propose a graph-based approach for fast computation of the actions of a composed rule. The approach has negligible effect on the performance of the composition operation itself, while resulting in the minimum number of actions to be performed in the data plane of switches;
4) We integrate our action composition algorithms in the CoVisor code base[2].

The rest of the paper is organized as follows: Section II describes the background and motivation for our approach. We present theoretical fundamentals and a model for action list composition in Section III. In Section IV, we detail efficient algorithms based on the model for the composition operators. Section V presents experimental results of these algorithms. We conclude with perspectives on our contributions in Section VI.

II. Background and Motivation

To set the scene, we first briefly present some features of SDN policy, and then review the parallel and sequential composition operations introduced in [14], [13], [19], [9]. Finally, we give examples to motivate our work.

A. SDN Policies

To fix ideas, one can think of OpenFlow [22] policies, although our work is general and not limited to OpenFlow. A policy is expressed as a set of prioritized *rules*. Each rule R is a 3-tuple $R = (p; m; a)$, where $R.p$ is the rule's priority, $R.m$ represents the match field patterns and $R.a$ is a sequential "program" (i.e. list) of the actions to be applied to packets matching the rule (see Figure 1).

Rule	Priority	Match	Actions
R_1	1	0000	*fwd(1)*
R_2	5	01**	*fwd(2)*
R_3	9	00**	*fwd(3)*
R_4	99	****	*fwd(4)*

Fig. 1. Example of policy as a rule table. Smaller priority values imply higher priorities.

The match fields in $R.m$ can, in all generality, consist of any number of adjacent packet bits (although they are usually limited to packet header fields) and ingress port. The set of match fields is the same for each rule in the policy, and their values can be any pattern including exact values, ranges (including prefixes), wildcards (matching any value), etc. If a packet potentially matches several rules, the rule with the highest priority is selected as the actual match, and the associated action list is applied to the packet. How a policy is implemented inside a switch (e.g. hardware table, pipeline of hardware tables, software hash, etc) is not relevant to the considerations of this paper.

We consider that actions are of three types: *modify* actions, whose effect is to modify packets or packet headers; *forwarding* actions, whose effect is to instantiate a packet on a port (i.e. forward the packet through the port); and *miscellaneous (misc)* actions, whose effect does not directly affect a packet (e.g. count actions, action list modification actions, etc.) Note that some of these misc actions have externally observable side-effects (such as actions modifying counters), while others do not (such as actions clearing the action list). To simplify, in this paper, we only consider counters associated with rules (one counter per rule) which count the number of packets for which the corresponding rule was a "hit" (and thus a `count` action simply increments such counter).

In this context, each rule of a policy is a function:

$$F(p) \rightarrow (p', port)^+ | d$$

where $(p', port)$ is a *forwarding pair* representing the packet p' appearing on port $port$, and d represents some statistics data side-effects. The $(\cdot)^+$-notation indicates that a rule can generate 0, 1 or more forwarding pairs for the given input packet p, depending on the packet's input port (which is part of the rule's matching pattern), and the packet itself. d is a positive integer (possibly 0) that represents the increment to be applied to the counter associated with the rule. Switches "implement" these functions by "executing" the actions associated with the rules[3].

From this, we can simply define the notion of *action list equivalence*: two action lists (i.e. two rule programs) are

[2]Our algorithms can be applied to other high-level programming frameworks very easily.

[3]More precisely, switches select the highest priority rule matching the packet and only execute the corresponding actions.

equivalent if and only if, for any packet p, $F_1(p) \equiv F_2(p)$. In other words, two action lists are equivalent, if they 1) produce the same forwarding pairs, and 2) count the same packets.

B. Composition Operators

The composition operators fall into two major categories: parallel composition and sequential composition. Here, we give a simplified overview for these composition operators and their compile algorithms presented in the prior art [13], [19].

Parallel Operator (+): The parallel operator compiles two policies into a single one which behaves as though packets were matched and processed by the two policies operating concurrently on their own copy of the traffic. For example, take a monitoring policy $Monitor$ that counts packets with source IP prefix 3.0.0.0/8 while dropping others. If a routing policy $Route$ forwards packets with destination IP 2.0.0.1 to port 1 and drops others (see Figure 2), then, with the parallel operator, we can generate a single policy $Monitor + Route$ shown in Figure 2.

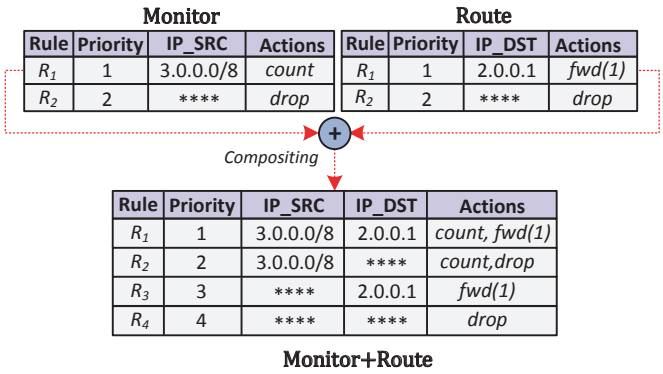


Fig. 2. Example of parallel composition, adapted from [16]

Next, we recall the existing compiler algorithms of the parallel operator using the example in Figure 2. To compile $Monitor + Route$, the compiler algorithms will calculate the cross product of rules from $Monitor$ and $Route$ as follows: any rule $m_i \in Monitor$ and $r_j \in Route$, m_i and r_j can generate a composed rule as long as $m_i.m \cap r_j.m \neq \emptyset$, using the intersection as its match fields and the concatenation of $m_i.a$ and $r_j.a$ as its action list. For example, consider m_1 and r_1 (the first rule in $Monitor$ and $Route$ respectively). As $m_1.m \cap r_1.m$ is {srcip=3.0.0.0/8, dstip=2.0.0.1}, they can generate a composed rule - the first rule in $Monitor + Route$.

Sequential Operator (>>): The sequential operator enables multiple policies to operate packets in series by combining those policies together. For example, suppose we have a load balancer policy LB that distributes traffic to two backend servers by rewriting their IP destination address while a routing policy $Route$ forwards packets based on their IP destination address (see Figure 3). Via the sequential operator, the composed policy will first rewrite the IP destination address and then forward these packets.

For the sequential composition of policies, the compiler algorithms compute the cross product of rules from the two

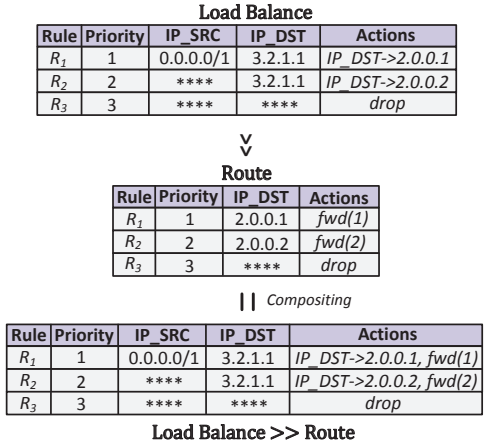


Fig. 3. Example of sequential composition.

policies ($LB >> Route$) as follows: apply the associated action list on the match fields of the rules from LB, and then check, for any rule $l_i \in LB$ and $r_j \in Route$, whether the intersection of $l_i.m$ and $r_j.m$ is empty or not. A composed rule is generated as long as $l_i.m \cap r_j.m \neq \emptyset$, through merging the match fields of $l_i.m$ and $r_j.m$ as the match fields, and concatenating $l_i.a$ and $r_j.a$ as the action list.

C. Motivating Examples

Let us first consider two policies, say P_1 and P_2, to be composed by parallel composition. From the very definition of parallel composition, the parallel composition of these policies should behave as though these policies operated in "parallel" on their own copy of the traffic. In other words, the packets generated by the parallel composition must be the union of the packets that would be generated by each policy operating on the traffic independently.

More formally, if $L_1(p)$, $L_2(p)$ and $L_{//}(p)$ denote the sets of forwarding pairs respectively generated by P_1, P_2 and the parallel composition of these policies, then

$$P_{//}(p) \equiv P_1(p) + P_2(p) \Rightarrow L_{//}(p) = L_1(p) \cup L_2(p)$$

It is trivial to prove that the parallel composition operator is commutative, that is that $P_1(p) + P_2(p) = P_2(p) + P_1(p)$, since $L_1(p) \cup L_2(p) = L_2(p) \cup L_1(p)$, confirming the intuition that the order in which the policies are composed should not affect the result of the parallel composition.

However, existing compositional systems all propose to construct the action list of a rule resulting from parallel composition as a simple concatenation of the action lists of each composed rule ($P_{//}(p) \equiv P_1(p) + P_2(p) \rightarrow a_{//}(p) = a_1(p) \circ a_2(p)$). Concatenation is obviously not commutative: if, for instance, $a_1(p)$ = {`dstip` $\leftarrow$ `8.0.0.2, fwd(2)`} and $a_2(p)$ = {`fwd(1)`}, then $a_1(p) \circ a_2(p)$ forwards the same packet (whose destination address has been changed to 8.0.0.2) on both port 1 and 2, while $a_2(p) \circ a_1(p)$ forwards the original input packet to port 1 and the packet with a modified

destination address to port 2. As parallel composition is a commutative operation, it therefore cannot be realized through simple action concatenation.

For sequential composition, which is not a commutative operation by definition, simple concatenation of action lists is also used. It is however, also easy to show that, while correct, concatenation of actions can lead to redundant actions. Indeed, consider, for instance, a_1 = {`vlan` ← `1`} and a_2 ={`vlan` ← `2`, `fwd(1)`}. $P_1 >> P_2$ yields $a_{>>}$ = {`vlan` ← `1`, `vlan` ← `2`, `fwd(1)`}. Conceptually, the first modification in the composed action list is redundant, leading to wastage in the resource constrained switch fast path[4].

We therefore see that simple concatenation for the composition of action lists cannot always preserve semantic equivalence and correctness, or achieve optimal operations in the data path. As a result, we conclude that action list composition, in the context of policy composition operators, needs to be revisited. We provide deeper analysis and solutions in the next few sections.

III. Action Composition Model

Essentially, actions are used in rules to transform input packets into output packets with specific properties, forward these output packets to output ports, as well as keep statistics on packets or rules. While other use of actions exists, such as circumventing a switch's lack of capabilities, it is the above mentioned observable results of actions that matter for compliance of the implemented policies.

The same is true for the composition operators: as long as the observable forwarding pairs and statistics comply with the intended compositional semantics, the result of the composition is correct.

A. Constructible Sequence and Graph-based Model

With the existence of `set/write` actions capable of setting any sequence of bits and/or fields to any specified value in the packet header, generating a packet with any specific header may seem trivial. However, this is not the case.

Indeed, the composition of policies is computed by the SDN hypervisor (a control plane component), using the policy rules, while the specific packet headers are only known by the switches (the data plane). In other words, the hypervisor can only rely on the rule matching patterns to represent packets, and the crux of the problem is that match patterns can contain "don't-care" bits (e.g. wild-cards, ranges, prefixes, etc.)

This is an issue, because once a part of a packet, corresponding to a match pattern containing "don't-care" bits, has been set to any specific value by a `set` action, there is generally no way to revert such change, as "don't-care" bits always match multiple values (see Figure 4).

The only way to revert a packet field, corresponding to a rule match field containing "don't-care" bits, is constructing switches that can copy and save the original field value from the input packet. However, current switch chipsets are not willing to support such actions for three reasons. First, recording packet values needs extra memory which is expensive in resource limited switch chips; second, enabling copy action causes race conditions because commodity switches usually process packets in parallel; third, each revert needs two memory copy operations (packet to memory and memory to packet), leading to a lower performance. Thus, packet fields fall into two categories:

[4]Any (unnecessary) operation in the data plane potentially leads to a decrease in forwarding rate.

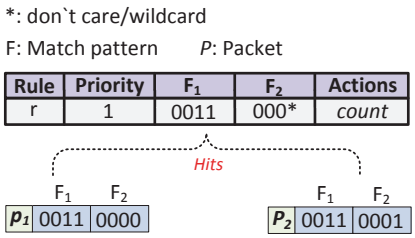


Fig. 4. Example of "don't-care" bits. F_2 in the rule contains one "don't-care" bit and thus matches two different values.

1) *Irreversible fields*: packet fields that 1) cannot be copied from the original (input) packet, *and* 2) correspond to match fields that contain "don't-care" bits in the policy rule.
2) *Reversible fields*: packet fields that either can be copied from the original (input) packet *or* that correspond to match fields specifying an exact (unique) value (no "don't-care" in the bit pattern of the field, the exact original value being thus available to the composing hypervisor).

Consequently, in the presence of changes to irreversible fields (see Figure 5), not every sequence of packets can be generated by a switch, from a given input packet. In fact, a set of output packets is said to be *constructible* from a given input packet if there exists a sequence (i.e. permutation) of those packets, starting with the input packet, such that no change to an irreversible field must be reverted to progress in the sequence. We now prove a fundamental theorem on constructible sequences of packets.

We call IC_i the set of irreversible fields that must change to generate output packet p_i from input packet p_{in}. Note that since changes to reversible fields can always be reversed (i.e. undone), reversible fields can safely be ignored in feasibility considerations.

Theorem 1. (Constructible Sequence Theorem)*: Given an input packet p_{in}, n output packets p_i and their set of irreversible field changes IC_i ($1 \leq i \leq n$), the sequence $< p_{in}, p_1, p_2, \ldots, p_n >$ is constructible iff $IC_1 \subseteq IC_2 \subseteq \ldots \subseteq IC_n$.*

Proof. We prove the forward direction by contradiction. Assume the sequence is constructible. Also assume that there exists an irreversible field if_k that changes to generate p_i, but does not change to generate p_j further in the sequence, that is $\exists if_k : if_k \in IC_i, if_k \notin IC_j$, with $i < j$.

Since $if_k \notin IC_j$, the value of if_k in p_j is the original value

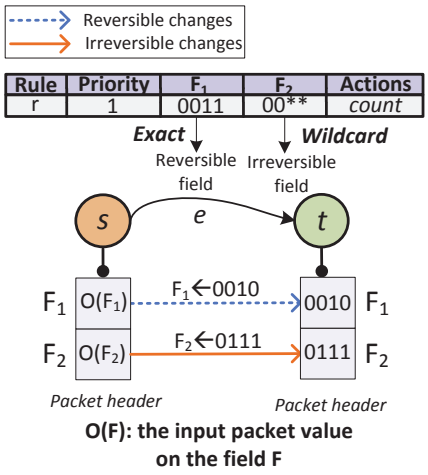


Fig. 5. Example of reversible and irreversible field changes.

of that field in p_{in}. Also, since $i < j$, p_j is constructed after p_i in the sequence, and this can only be possible if the change to irreversible field if_k, that was necessary to generate p_i has been reversed to generate p_j. This is a contradiction, since if_k is an irreversible field. We therefore have that in a constructible sequence, $(i < j, \forall if_k : if_k \in IC_i) \Rightarrow if_k \in IC_j$, which implies that $IC_i \subseteq IC_j, i < j$.

We prove the reverse direction by induction. **Base case**: by definition of IC_k, any packet p_k can be constructed from p_{in} by changing the irreversible fields in IC_k (along with possibly changes to some reversible fields). In particular, p_1 can always be generated from p_{in} by changing the (irreversible) fields in IC_1 (such operation is denote $p_{in} \rightarrow_{IC_i} p_1$). **Inductive case**: assume the prefix subsequence up to packet p_k ($< p_{in}, p_1, p_2, \ldots, p_k >$) is constructible. We show that p_{k+1} is constructible (can be generated) from p_k, given that $IC_k \subseteq IC_{k+1}$. Indeed, $IC_{k+1} = (IC_k \cap IC_{k+1}) \cup (IC_{k+1} \setminus (IC_k \cap IC_{k+1}))$. But since $IC_k \subseteq IC_{k+1}$, we have that $IC_k \cap IC_{k+1} = IC_k$, so that $p_{in} \rightarrow_{IC_k \cap IC_{k+1}} p_k$. This means that p_k can be generated as a step in the construction of p_{k+1}. From this step, the remaining changes in IC_{k+1}, that is all the changes in $IC_{k+1} \setminus (IC_k \cap IC_{k+1})$ can be applied to yield p_{k+1} ($p_k \rightarrow_{IC_{k+1} \setminus (IC_k \cap IC_{k+1})} p_{k+1}$). We therefore have $p_{in} \rightarrow_{IC_k \cap IC_{k+1}} p_k \rightarrow_{IC_{k+1} \setminus (IC_k \cap IC_{k+1})} p_{k+1} = p_{in} \rightarrow_{IC_{k+1}} p_{k+1}$. □

When an SDN hypervisor is composing policies, it does not generally know the exact values of the fields of the packets that will hit the resulting rules. Still, it can "simulate" the effects of applying the actions associated with the rules (according to the composition operators used), so that it can "compute" the packets, in terms of which input packet fields get modified or not, and on which ports these packets get forwarded. The discussion and results describe above therefore suggest that the problem for the hypervisor is thus to find the right sequence for generating the output packets, given that as soon as an output packet has been constructed, it can simply be forwarded to the correct ports by issuing appropriate `forward` actions.

A convenient way to model the process of constructing packets is thus as a graph, where vertices represent each unique packet in the process (that is the input packet and each output packet to be generated), and where there is an oriented edge between two vertices if a series of actions can transform the source packet into the destination packet. The important thing to remember, is that reversible packet fields can always be changed to any value in any order, while irreversible fields can only be set to specific (known) values, but cannot be reverted to their unknown original (input) value. The resulting graph is thus not a "full mesh" (since some packets cannot be constructed from others). Each edge in the graph can then be labelled with the set of packet modification actions needed to actuate the transformation from the source packet to the destination packet (see Figure 6).

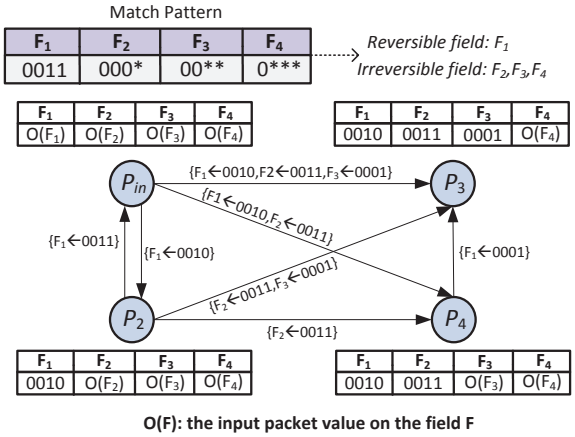


Fig. 6. Example of graph-based action composition. $IC_{p_{in}} = IC_{p_2} = \emptyset$, $IC_{p_3} = \{F_2, F_3\}$, $IC_{p_4} = \{F_2\}$. An oriented edge from p_i to p_j exists *iff* $IC_{p_i} \subseteq IC_{p_j}$. A path starting from the input packet $p_i n$ and visiting each vertex is $p_{in} \rightarrow p_2 \rightarrow p_4 \rightarrow p_3$. The corresponding action list is $F_1 \leftarrow 0010, F_2 \leftarrow 0011, F_1 \leftarrow 0001$.

With such a graph, generating the required packets, and computing the associated action list, reduces to finding a Hamiltonian path [23], starting at the input packet, if such path exists. Indeed, a Hamiltonian path through a graph visits each vertex exactly once, corresponding to every output packets being generated.

However, the Hamiltonian path problem is known to be NP-complete [23], [24]. In section IV, we discuss algorithms to find such a path, while aiming to minimize the number of actions required to actuate the construction of output packets.

B. Misc Action Considerations

A misc action associated with a rule counts the number of packets for which the corresponding rule was a "hit". Let us consider two policies P_1 and P_2 that need to be composed. Suppose $r_1 \in P_1$ has a misc action to count $C(r_1)$ the number of packets that hit r_1. After composition of P_1 and P_2, we still need to get $C(r_1)$ from the composed policy.

Let $S(r_1, P_2)$ denote the set of composed rules that contain the semantic of r_1, i.e., $S(r_1, P_2) = \{r_1.m \cap t_i.m | r_1.m \cap t_i.m \neq \emptyset, t_i \in P_2\}$. For each composed rule $s_i \in S(r_1, P_2)$,

we associate one misc action to count the number of packets that hit s_i. We then have the following Theorem for the restoration of $C(r_1)$ from the composed policy.

Theorem 2. (QUERY STATISTICS)*: Given two policy P_1 and P_2, policy M is composed of P_1 and P_2. The counter $C(r_1)$ associated with the rule $r_1 \in P_1$ can be computed as:*

$$C(r_1) = \sum C(s_i)$$

where $s_i \in S(r_1, P_2) \subseteq M$ and $S(r_1, P_2) = \{r_1.m \cap t_i.m | r_1.m \cap t_i.m \neq \emptyset, t_i \in P_2\}$.

Proof. On the one hand, for any packet that hits r_1, it can hit at least one rule of $S(r_1, P_2)$: the rule composed by r_1 and the `default` rule of P_2. On the other hand, due to the priorities of composed rules, any packet can hit no more than one rule of $S(r_1, P_2)$. As such, any packet that hits r_1 can hit exactly one rule of $S(r_1, P_2)$. In other terms, the number of packets that hit r_1 is equal to the number of packets that hit the rules of $S(r_1, P_2) \subseteq M$. □

IV. ACTION COMPOSITION ALGORITHMS

We showed in section III that the problem of finding a constructible sequence of packets to implement the composition of policies reduces to finding a Hamiltonian path in a graph. While this problem is generally NP-complete, Theorem 1 states a fundamental property of such sequences that can be exploited to efficiently find such sequence.

Indeed, Theorem 1 shows that, in a constructible sequence, changes to irreversible fields must be applied "incrementally", due to the "nesting" of the set of irreversible fields that have changed (compared with the input packet), from one packet in the sequence to the next; in other words, packets further in the sequence, can only be constructed by either "adding" more changed irreversible fields or changing again (to specific know values) some of these fields, compared with earlier packets in the sequence.

This observation leads to a very simple, straightforward and efficient algorithm (see Algorithm 1) to not only test for the existence of a constructible sequence, but also obtain one such sequence of packets (if it exists).

All we need to do is to represent all the irreversible fields in a rule as a bitmap. Remember that what makes a header field irreversible is the presence of "don't care" bits in the pattern of the rule representing that field and the lack of switch capability to save the original value of this header field in the input packet, both properties being known to the compositional hypervisor. Then for each desired output packets (again, these are know to the hypervisor), set to 1 the bits corresponding to changed irreversible fields (lines 1 to 4). Then sort the "output packets" by the *number* of bits set in the bitmap (line 5), because irreversible field changes must be applied incrementally. Then sweep across the ordered packets, checking if `bitmap(k) & bitmap(k+1) == bitmap(k)`, which is equivalent to checking that the set of irreversible field changed in one packet is completely contained in the set of irreversible fields changed in the next packet (as required by Theorem 1). If this test succeeds for each consecutive pair of packets, then not only a constructible sequence of packets exists, but the ordered packets is one such sequence (lines 6 to 11).

Algorithm 1: SIMPLESEARCH(p_{in}, $\{p_{out}\}$, $\{IF\}$)

Input: p_{in}: input packet
Input: $\{p_{out}\}$: set of (unique) output packets
Input: $\{IF\}$: set of irreversible fields in the rule
Output: path: Hamilton path "vector" ("empty" if no such path exists)

```
1  path ← newEmptyVector();
2  for p ∈ {p_out} do
3      bm ←BitMap(p_in, p, {IF});
4      path.append((p, bm));
5  sort(path, ByNumberOfBitSet);
6  thisP ← path.first();
7  while (nextP ← path.next()) ≠ NULL do
8      if thisP.bm & nextP.bm ≠ thisP.bm then
9          return newEmptyVector();
10     thisP = nextP;
11 return path;
```

The complexity of this algorithm, given n output packets to generate, is $O(n)$ for generating the bitmaps; $O(n \lg n)$ for sorting; and $O(n)$ for testing the inclusion relation. Therefore the overall complexity is $O(n \lg n)$.

From the returned sequence of packets (if it exists), the compositional hypervisor can compute the action list for the corresponding (composed) rule by simply concatenating the modify actions required to generate each packet in the path, from the preceding packet, and issuing the required forwarding actions whenever the desired packets have been generated.

While Algorithm 1 finds a constructible sequence of packets if such sequence exists, this sequence may not be optimal in terms of the number of actions required to generate the sequence. This is because packets that have the *same* set of modified irreversible fields (and thus only differ from each other by different sets of modified *reversible fields*) can appear in *any relative order* in the sequence.

See, for instance, packets P_3 and P_4 in Figure 7. The total cost of the path is 6 (① + ② + ③ + ④). But there is another constructible sequence of packets, obtained by exchanging packet P_3 and P_4 in the packet sequence, with reduced cost 5. This is because the cost from P_5 to P_4 is 1 ($F_3 \leftarrow 0001$) while P_4 to P_3 requires 2 modification operations ($F_1 \leftarrow 0011, F_3 \leftarrow 0011$). The reason why we can change the order of P_3 and P_4 to get a lower cost path is that they contain identical sets of modified irreversible fields, i.e. $IC_3 = IC_4$.

While Algorithm 1 actually worked on an *implicit* representation of the graph model for packets described in Section III, finding optimal sequences will require an explicit representa-

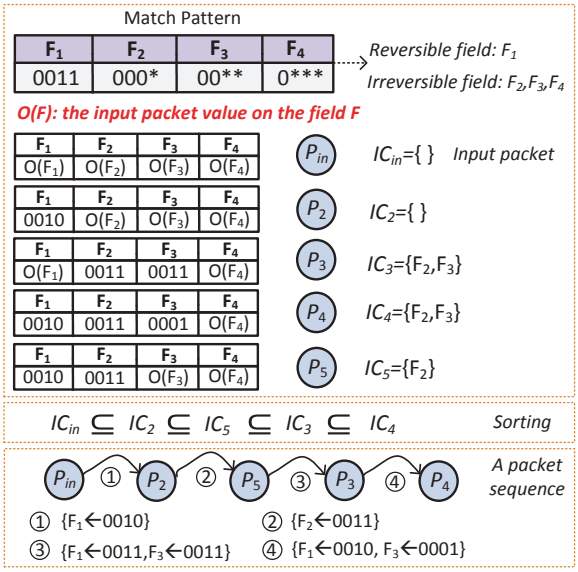


Fig. 7. Example of a Hamilton path. The packets sequence sorted by the number of irreversible changes provides one Hamilton path.

tion of this graph.

The graph for packet generation as described in Section III would have a directed edge between two packets if no modified irreversible field has to be reverted to its original value (in the input packet) to go from the "source" vertex to the "destination" vertex. However, this is far too many edges: indeed, because of the transitivity of the "subset" relationship (i.e. "contain" operations) required of the modified irreversible field subsets of the packets in a constructible sequence (Theorem 1), a sub-sequence $P_1 \rightsquigarrow P_2 \rightsquigarrow P_3$ [5], would also imply one directed edge $P_1 \rightsquigarrow P_3$. However, the $P_1 \rightsquigarrow P_3$ edge is completely useless, because it will never be part of a Hamiltonian path in the graph: a sequence can never go back to P_2 from P_3, as this would mean reverting (at least) one irreversible change.

The output of the simple Algorithm 1 can here help avoid generating these useless edges in the graph, and thus reduce the space to be searched for optimality. Indeed, this simple algorithm outputs packets ordered by their number of modified irreversible fields. Any sub-sequence of adjacent packets with the same number of such modifications thus forms a group of packets whose order can be changed while still conserving a constructible sequence. This is because packets in each group form a "local full-mesh", and they only differ from each other by modifications to reversible fields. The simple algorithm therefore also gives the sequence of groups, and there thus only needs to be an edge from each packet in a group, to each packet in the following group in the sequence (see Figure 8).

While finding an optimal path (in terms of the number of actions needed) in such graph is still an NP-complete Hamiltonian path search, we argue that in practical scenarios, the number of distinct output packets to be generated will be

[5]We suppose $IC_1 \subseteq IC_2 \subseteq IC_3$. $\forall i \in [1,3]$, IC_i corresponds to P_i.

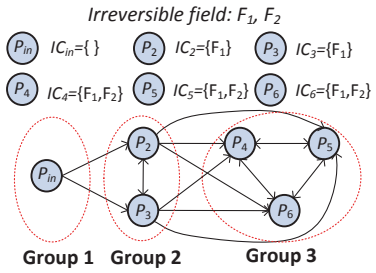


Fig. 8. Example of packet grouping.

kept relatively low (so the number of vertices in the graph will be small). Furthermore, this graph only contains edges that potentially belong to a constructible sequence (so the number of edges has been reduced to a minimum). Because of this "reduced" search space, we believe that a brute-force algorithm, enumerating all the (Hamiltonian) paths in the graph is a plausible solution to the optimal Hamiltonian path finding problem at hand.

Nevertheless, should the search space become too big, the compositional hypervisor can always decide to use a heuristic algorithm (such as one based on a greedy approach) instead, to trade running time for potential deviation from optimality[6].

V. Implementation and Evaluation

We have implemented our model and the related algorithms in CoVisor [13]. Using this implementation we evaluate its performance.

More specifically, we replaced the core logic of action lists composition for both the parallel and sequential operators. Note that we implemented three Hamilton path searching algorithms: the simple algorithm (Algorithm 1), the brute-force (a.k.a. enumeration) algorithm and a greedy algorithm, picking the less weighted edge whenever a choice is available when searching for the Hamilton path: suppose the last added vertex in the path is v, then the next vertex in the path is u=argmin$_{u \in U(v)} W(v,u)$, where $W(v,u)$ is the weight (i.e. the number of modification actions) of the edge from v to u and $U(v)$ is the set of destination vertices of edges from v. This greedy algorithm works, because we ensure that the graph only contains edges that are potentially part of a Hamilton path (see Section IV).

A. Experimental Setup

We deployed our implementation on an octo-core Intel®Xeon®E5506 CPU, clocked at 2.13GHz. The machine is equipped with 16GB RAM and runs 64-bit Ubuntu Linux 10.04.3. We used two rulesets for our experiments:

1) D1 (real-life policies): L3 Router [26] and L3 Firewall [27].
2) D2 (synthetic policies): rules are generated associated with multiple types of actions (e.g. modification, for-

[6]As an extreme case, the hypervisor could even choose to use the output of the simple algorithm.

warding and misc actions) to reflect more dynamic, complex scenarios.

Each rule of D1 contains one forwarding action. Each rule of D2 on the other hand contains multiple modification/forwarding actions. To generate modification actions in D2, we randomly select one packet header field as the field that is modified by the action, whose value after modification is also randomly assigned. The number of distinct output packets for each rule is controlled through forwarding actions. In the experiment, an action list can generate no more than 10 distinct (different) output packets for one input packet – we believe this value to represent an unrealistic value, chosen to illustrate absolute worst case scenarios. The match pattern for IP address is prefix-based, while for other match patterns (like port, MAC address, vlan), we use exact match.

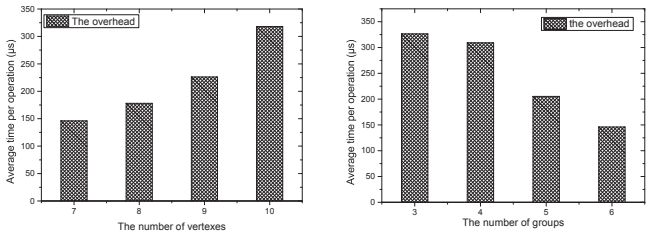
We are interested in four aspects of performance: 1) the computation time; 2) factors that affect the computation time; 3) contribution of the various components to the computation time; 4) comparison between the three path search algorithms in terms of computation time and optimality.

B. Experimental Results

TABLE I
COMPUTATION TIME OVER TWO POLICIES (IN μS).

	average	minimum	maximum
D1	85	72	95
D2	249	125	380

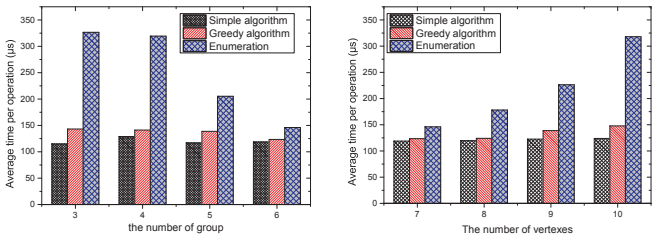
The average, minimum and maximum computation time of the enumeration algorithm are reported in Table I. The action lists of any rule in D1 can be computed within 95 μs. Computation of action lists for rules in D2 takes a longer time and the average time is around 250 μs. This is because rules in D2 have more complex actions and can generate more distinct output packets. Nevertheless, the computation time is relatively small, showing that our approach is practical.



(a) (b)

Fig. 9. Variation of computation time with two factors: (a) the number of vertices in a group, (b) the number of groups.

The computation time depends on both the number of vertices in groups and also the number of groups in our graph-based model. We first select the actions from $D2$ that have 6 groups. Figure 9(a) plots the computation time when varying the number of vertices. As expected, a larger number of vertices leads to a higher computation time. But, even with 10 vertices, the computation time is still within 320 μs. We then select the actions from $D2$ that have 7 vertices. The computation time with different number of groups is reported in Figure 9(b). A larger number of groups leads to a smaller number of vertices per group. Given that the permutation within each group is one of the major contributors on computation time, a smaller number of vertices in each group in turn results in lower computation time.



(a) (b)

Fig. 10. Comparison of three algorithms in terms of computation time: (a) varying the number of groups. (b) varying the number of vertices.

We then compare the three algorithms for the Hamilton path search in terms of computation time. Figure 10 shows the computation time of the three algorithms, where we vary the number of groups while fixing the number of vertexes (Figure 10(a)), and vary the number of vertexes while fixing the number of groups (Figure 10(b)). Since the enumeration and greedy algorithms use extra optimization (necessitating the output of the simple algorithm to generate a "reduced" graph, see Section IV), they requires to use more computation time. Compared with enumeration algorithm, the greedy one can save up to 50% of the computation time, and is less relevant to the number of vertices and the number of groups.

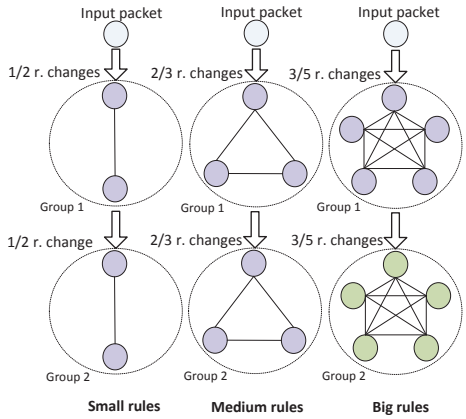


Fig. 11. The three scenarios. r. represents reversible fields.

Finally, we evaluate the amount of actions in the Hamilton path generated by the three algorithms. To this end, we construct three scenarios based on D2 (see Figure 11). In all three scenarios, one input packet would generate two groups of outputs. In the *small rules* scenario, each group contains two packets which change 1 (out of 2) reversible field; in the *medium rules* scenario, each group has three packets which change 2 (out of 3) reversible fields; in the *big rules* scenario,

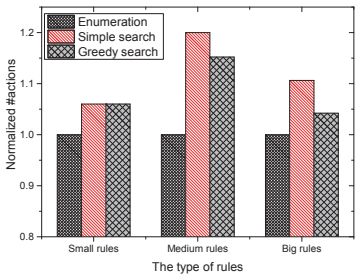


Fig. 12. Number of actions generated by the three path search algorithms, normalized by the number of actions in the path generated by the enumeration algorithm.

each group has five packets which have 3 (out of 5) reversible fields. We apply the three algorithms on each scenario and plot the number of actions in the path generated by each algorithm in Figure 12. We can see that, compared with the enumeration algorithm (which is optimal), the simple search algorithm generates up to 20% more actions, and the greedy algorithm incurs up to 15% more actions.

In summary, given that the composition of policies is performed in servers (like controllers) other than switches themselves, we believe the enumeration algorithms is more applicable in practice in order to obtain optimal results.

VI. CONCLUSION

Policy composition has been emerging as a powerful and important tool for facilitating the creation and deployment of complex network applications. As the developer or network administrator requesting such composition may not master, or even want to know, the details of each policy component being composed, it is of paramount importance that compositional operators be supported in as much a transparent and efficient manner as possible. Previous work introduced important headways in this direction by proposing efficient techniques to compute the matching patterns for composed rules. Our work complements this by tackling the problem of correct and efficient action list computation, another important component of policy rules.

In particular, we formalize an action composition model, and prove a feasibility condition on the composition of rule actions. We abstract the action composition as a Hamilton path search problem in a directed weighted graph, while exploiting fundamental properties specific to the resulting graph to compute solutions, to this otherwise NP-complete problem, efficiently. We show that our approach is not only correct, but also efficient.

ACKNOWLEDGMENTS

We thank the IFIP Networking reviewers for their insightful feedback. This work is supported in part by National High Technology Research and Development Program of China (Grant No. 2015AA016101 and 2015AA010201), National Natural Science Foundation of China (Grant No. 61502458 and 61502462) and Beijing Municipal Natural Science Foundation (Grant No. 4162057).

REFERENCES

[1] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, 2009.
[2] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.
[3] "The hp sdn app store." http://h17007.www1.hp.com/us/en/networking/solutions/technology/sdn/devcenter/#sdnAppstore.
[4] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.
[5] "Opendaylight." http://www.opendaylight.org/.
[6] "Ryu openflow controller." http://osrg.github.io/ryu/.
[7] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.
[8] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. Kleinberg, E. G. Sirer, and N. Foster, "Merlin: A language for provisioning network resources," in *ACM CoNEXT*, 2014.
[9] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," in *ACM SIGPLAN Notices*, vol. 46, pp. 279–291, ACM, 2011.
[10] C. Monsanto, N. Foster, R. Harrison, and D. Walker, "A compiler and run-time system for network programming languages," *ACM SIGPLAN Notices*, vol. 47, no. 1, pp. 217–230, 2012.
[11] Y. T. Chaithan Prakash, Jeongkeun Lee and J.-M. Kang., "Pga: Using graphs to express and automatically reconcile network policies," in *Proceedings of the 2015 ACM conference on SIGCOMM*, ACM, 2015.
[12] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, and R. Clark, "Kinetic: Verifiable dynamic network control," 2015.
[13] X. Jin, J. Gossels, J. Rexford, and D. Walker, "Covisor: A compositional hypervisor for software-defined networks," in *Proc. USENIX NSDI*, 2015.
[14] A. Dixit, K. Kogan, and P. Eugster, "Composing heterogeneous sdn controllers with flowbricks," in *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, pp. 287–292, IEEE, 2014.
[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
[16] C. Monsanto, J. Reich, N. Foster, J. Rexford, D. Walker, *et al.*, "Composing software defined networks.," in *NSDI*, pp. 1–13, 2013.
[17] N. Foster, A. Guha, M. Reitblatt, A. Story, M. J. Freedman, N. P. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger, *et al.*, "Languages for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 128–134, 2013.
[18] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker, "Netkat: Semantic foundations for networks," *ACM SIGPLAN Notices*, vol. 49, no. 1, pp. 113–126, 2014.
[19] X. Jin, J. Rexford, and D. Walker, "Incremental update for a compositional sdn hypervisor," in *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 187–192, ACM, 2014.
[20] "The opensource code of covisor." https://github.com/CoVisor/CoVisor.
[21] "The code of frenetic language.." http://frenetic-lang.org/pyretic/.
[22] "Openflow switch specification." https://www.opennetworking.org/.
[23] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd ed., 2001.
[24] A. A. Bertossi, "The edge hamiltonian path problem is np-complete," *Information Processing Letters*, vol. 13, no. 4, pp. 157–159, 1981.
[25] "Openflow switch specification." https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf.
[26] "Routereview." http://www.routeviews.org/.
[27] "The rules set of evaluation packet classification." http://www.arl.wustl.edu/~hs1/PClassEval.html.

Anomaly-Free Policy Composition in Software-Defined Networks

Mohsen Rezvani, Aleksandar Ignjatovic, Maurice Pagnucco and Sanjay Jha
School of Computer Science and Engineering, UNSW Australia
{m.rezvani,a.ignjatovic,m.pagnucco,sanjay.jha}@unsw.edu.au

***Abstract*—Software Defined Networking (SDN) provides considerable simplification of design and deployment of various network applications for large networks. Each application has its own view of network policy and sends its policy to a network hypervisor in which a composed policy is generated from the application policies and deployed into the data plane. A significant challenge for the hypervisor is to detect and resolve both intra and inter policy anomalies during the policy composition. However, current SDN compilers do not consider the policy anomalies well and generate large number of unnecessary rules for the data plane. This leads to a considerable inefficiency in both policy composition and policy deployment. In this paper, we propose a novel framework for policy composition in a SDN hypervisor which takes into account both inter and intra policy anomalies. Moreover, we augment the framework with an efficient insertion transformation mechanism which allows the applications to issue rule insertion and priority change updates. Our evaluation shows that our method is several orders of magnitude more efficient than the state of the art in both policy composition and compiling the rule insertion updates.**

I. Introduction

Software Defined Networking (SDN) is transforming traditional network architectures to more flexible and programmable platforms by decoupling the control logic from the forwarding (data) layer. A logical SDN architecture includes three distinct layers: application, control and forwarding (data) [1]. Network applications at the top of this multi-layer architecture can define network policies based on a global view of the network provided by software-based controllers in the control layer. The controllers enforce network policies in the data layer by translating application defined policies into low-level and identifiable rules in network devices. The OpenFlow protocol [2] is one of the earlier and more popular communication standards between the control and data layers.

The SDN multi-layer architecture allows multiple applications or even multiple administrators to specify the network policy simultaneously. Each application can take advantage of the global network view to effectively define its network policy as a sequence of OpenFlow rules. The controller then, as a *hypervisor* integrates the policies received from different applications based on a policy composition strategy. The strategy specifies how to use three common binary operators: *parallel* (two policies can be applied at the same time), *sequential* (the second policy is processed after the first one) and *override* (the second policy is applied only on the traffic which is not matched by the first policy) to combine the policies [3]. As a result of such policy composition, the controller generates a set of prioritized OpenFlow rules, called *composed policy* and installs such a policy into the data plane (as shown in Fig. 1).

ISBN 978-3-901882-83-8

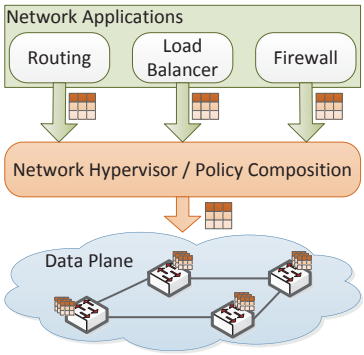


Fig. 1: Policy composition in a SDN hypervisor.

A naive policy composition method is to recompute and reinstall the composed policy every time an application updates its policy. Jin et al. [4] recently proposed *CoVisor* which incrementally updates the policy without shifting existing rules. CoVisor limits the policy updates to only two operations: *add* a new rule and *delete* an existing rule. However, in many applications, such as firewalls, an administrator can *insert* a new rule in the middle of existing rules. Such insert rule operation can lead to a shift of many rules if there is no empty space for the priority of the new rule [5]. Accordingly, applying a rule insertion in CoVisor needs in average $\frac{n}{2}$ delete and $\frac{n}{2}$ add operations in the base policy, where n is the number of rules in the policy. Note that a composition operator, such as parallel operator, after such many updates in the application policy may generate $O(n^2)$ updates in the composed policy.

Another important challenge in SDN policy management is to detect and resolve policy anomalies, such as redundancies and conflicts, in the application policies and then combine them to make an anomaly-free composed policy. This not only reduces the number of rules in the application-level policies [6], but also considerably improves the efficiency of the policy composition as it prevents cascading the anomalous rules into the composed policy. Although the anomaly detection within an individual application policy (intra-policy anomalies) has been well investigated in the literature [6], [7], [8], [9], the SDN multi-layer architecture makes the anomaly detection more difficult. This is because existing anomalies among policies (inter-policy anomalies) must be considered. Our experiments show that the policy anomaly detection is more significant in a SDN hypervisor as its policy composition can quadratically propagate the existing intra-anomalies into the composed policy. This can result in deploying many unnecessary rules into the data plane which leads to not only inconsistency in the network policy but also significant inefficiency due to deployment of these rules.

To address the above challenges, we first propose a novel anomaly-free policy model which help us to efficiently detect and resolve anomalies for policy updates. We maintain a separate model for each application-level policy. Then, we define the policy composition operators over our policy model which helps us to generate a model for the composed policy. Thus, the results of policy composition is a policy model which is also an anomaly-free model. We then propose a straightforward algorithm to translate the model to low-level OpenFlow rules. Moreover, we leverage our policy model to efficiently translate the rule insertion updates received from the application policies. The policy model efficiently emits the sequence of prioritized rules which reduces the complexity of both anomaly detection and policy composition.

In summary, we make the following contributions.

- We propose a new model for OpenFlow-based policies which allows to efficiently obtain the dependencies between OpenFlow rules;
- We develop a formal mechanism to detect and resolve both intra and inter-application policy anomalies which leverages our OpenFlow policy model to reduce the complexity of the detection process;
- We develop a new algorithm to incrementally compose inserting updates which eliminates unnecessary shifting in both individual and composed policies.

We provide a comparative evaluation of the performance of our algorithms with the state of the art in SDN policy composition. The results show that our method significantly improves the efficiency of the policy composition by reducing the update length several orders of magnitude compared to proposed method in [4]. Moreover, the proposed insertion translation considerably increases the performance of the naive approach for handing insertion updates.

The rest of this paper is organized as follows. Section II presents the related work. Section III describes the details of our policy model. Section IV presents our anomaly-free policy composition system. Section V describes our experimental results. Finally, the paper is concluded in Section VI.

II. Related Work

Anomaly detection in traditional access control policies, such as firewalls, has been extensively studied in the research community [10], [6], [7], [8], [11]. Al-Shaer and Hamed present a set of algorithms to discover simple pairwise anomalies in centralized and distributed firewall rules [6]. Inconsistencies and inefficiencies among multiple rules are treated in [7], [9]. Adao et al. [11] propose *Mignis*, a declarative policy language to specify a Linux firewall, Netfilter configurations. The Mignis tool is tightly integrated with Netfilter and is hard to use for the OpenFlow policies.

Several SDN policy languages, such as Frenetic [12], NetKAT [13] and Pyretic [3], have been proposed. They use different batch mechanisms for policy composition which make a large number of updates for each update in an application policy. Wen et al. [5] introduced an incremental policy update for Frenetic policies. The proposed method maintains a dependency graph and scattered priority distribution for each policy. Our method, instead of a using a dependency graph, maintains an anomaly-free model which efficiently handles the priority updates, such as insertion, without maintaining any priority distribution. Recently, Jin et al. [4] proposed CoVisor which employs a simple algebra for priority assignment in an incremental policy composition. However, CoVisor neither considers policy anomalies nor rule insertion updates. While our method uses the algebra proposed in CoVisor, it also takes into account both policy anomalies and insertion updates.

Han et al. [14] proposed a multi-layer policy management framework for SDNs. However, the proposed method for intra-policy anomaly resolution makes the policy enforcement a nondeterministic task which can affect the rules' semantics and change the intention of policy definition [8]. Moreover, the proposed method for inter-policy anomaly detection does not consider the priority assignment in the incremental policy composition. Dwaraki et al. [15] proposed *GitFlow*, a conflict-free flow repository management for SDNs. However, GitFlow considers neither the total conflicts in the application policies nor the conflicts during the policy composition. Shin et al. [16] present FRESCO, a framework for developing and deploying network security applications for OpenFlow networks. However, its conflict detection module only considers simple pairwise rule conflicts in the data plane level. Prakash et al. [17] recently proposed policy graph abstraction (PGA), a high level policy graph abstraction which provides a conflict detection and resolution mechanism. However, PGA is a whitelisting model while the OpenFlow policies in an SDN hypervisor can contain blocking rules. Smolka et al. [18] proposed a fast compiler for NetKAT which introduces forwarding decision diagrams (FDDs) to improve the efficiency of BDDs for encoding the packet headers. Although our experiments show a promising performance with BDDs, in our framework one can also use FDDs for conflict detection and resolution.

III. SDN Policy Model

We assume that each application defines its policy as a set of flows defined in OpenFlow specification [19]. In this paper, the term *rule* refers to a flow in OpenFlow specification.

A. OpenFlow Rule

Without loss of generality, we represent an OpenFlow rule r with three components: 1) a *priority*, denoted as $r.priority$, is a non-negative integer value used for matching precedence of the rule within a policy; 2) *rule match*, denoted as $r.match$ which is a set of *matching fields* for specifying a set of packets; and 3) *actions*, denoted as $r.actions$ which is a set of instructions to apply on the packets. A rule r defines how a set of packets specified in $r.match$ is treated in the network.

In the basic model, the matching fields represent information about source and destination of the matched packets. Thus, we can simply divide these fields into two sets where each of them is called a *peer*. A combination of these fields defines the source (destination) of the matched packets and is called *source (destination) peer*, denoted as $r.speer$ ($r.dpeer$) for a rule r. In other words, source (destination) peer is a generalization to specify the source (destination) sides of the matched packets. Specifically, $r.speer$ ($r.dpeer$) is a combination of matching fields including source (destination) MAC, source (destination) IP, source (destination) port, and protocol. A matching field x for a peer p is denoted as $p[x]$. Note that some fields in OpenFlow, such as protocol and VLAN identifier, are neither assigned to source nor destination as they are common

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

for both peers. Thus, we repeat their values in the specification of both peers if a rule is defined by these fields.

In order to model a peer of a rule, we use the idea of binary representation of packet headers proposed in [9], [20], [21]. To this end, we first encode each matching field as a bit stream and a peer is then represented as a bit stream obtained from a combination of bit streams of its matching fields. In the basic model, the representation of a matching field in a peer contains: a MAC address as a stream of 48 bits, a network address as a stream of 32 bits, a port as a stream of 16 bits, and a protocol as a with of 8 bits. Thus, a peer is encoded as a stream of $48 + 32 + 16 + 8 = 104$ bits.

Using the above encoding, a matching field x in a peer p is represented as a propositional logic formula, denoted as $p[x].formula$. We define n variables to make the formula from a stream of n bits. The formula is conjunction of the variables for every bit set 1, its negation for every bit set zero, and nothing for every bit set *don't care* [9]. Now rule r is represented by a conjunction of the formulas obtained for its matching fields. Thus, $r.match.formula = \bigwedge_{x \in r.match} x.formula$ and the match formula contains at most $104 * 2 = 208$ variables.

A rule match can be defined based on conjunction of matching fields. All fields but network addresses define either all values using a *wildcard* or only one specific value. A network address is represented by a CIDR domain which defines an arbitrary set of IP addresses. Thus, a matching field can be specified by a set of values. Now we show that all the fields hold the DISJOINTORSUBSET property which is employed to propose a hierarchy representation of different matching fields in a policy. Note that OpenFlow proposes a bitmask feature which allows matching single bits of a filed [19]. DisjointOrSubset holds when there is no such bitmask in the fields. Supporting the bitmask is our future work.

Proposition 1 (DISJOINTORSUBSET)**.** *Each two values in a matching field are either disjoint or one is a subset of the other. More formally, assume that p_1 and p_2 are two peers and x is a matching field defined according to our OpenFlow rule model. The following relation holds:*

$$\forall p_1, p_2 : (p_1[x] \cap p_2[x] = \emptyset) \vee (p_1[x] \cap p_2[x] = p_1[x]) \vee (p_1[x] \cap p_2[x] = p_2[x]). \quad (1)$$

Proof. Since all matching fields but network address define either wildcard or only one specific value, it is obvious that they satisfy the property. A network address field is specified as a masked CIDR domain which contains an IP address and a *mask* which is an integer value in the range of [0,31]. Clearly, if two network domains contain non-overlapping IPs, they are disjoint and otherwise one with smaller mask is a subset of the other one. Thus, all the matching field holds the property. □

B. OpenFlow Policy

Our idea for modeling a network policy is the fact that the main source of inefficiency in a SDN hypervisor is the set operations, such as union and intersection of match rules, needed during both policy composition and anomaly detection. We propose a graphical model to represent an OpenFlow policy which is inspired by PGA [17]. Since our solution is deployed within a SDN hypervisor (as shown in Fig. 1), our policy model is required to 1) specify prioritized rules while PGA only supports a set of rules without any priority; 2) specify different actions according to OpenFlow specification [19] which includes conflicting actions, such as *Forward* and *Drop*, while PGA is limited to a whitelisting policy containing only permit rules; and 3) incrementally compose policies and generate minimum updates for deploying into the data plane, while PGA supports batch policy composition.

An OpenFlow policy is a set of rules $P = \{r_1, r_2, \dots r_n\}$, where $r_i = (r_i.priority, r_i.speer, r_i.dpeer, r_i.actions)$, $(1 \leq i \leq n)$ represents the i^{th} rule in the set. Now we introduce the Policy Semantics Graph (PSG) to model an OpenFlow policy.

Definition 1. ***(Policy Semantics Graph)*** *A PSG is a directed graph, generated from an OpenFlow policy P. The vertices are the peers in the rules in P. Also, there is an edge between two vertices corresponding to peers p_1 and p_2 if and only if $\exists r \in P, r.speer = p_1 \wedge r.dpeer = p_2$. An edge represents a rule in the policy and consists of the priority and actions of the corresponding rule.*

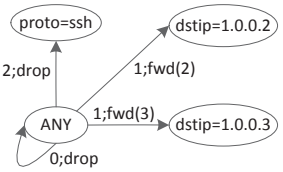
Corollary 1. *A PSG model has no multiple edges between any two vertices (thus a PSG is not a multigraph).*

Proof. Follows directly from the fact that PSG has one and only one edge corresponding to each rule in the policy. □

Corollary 1 shows that PSG is a replica-free policy model. Thus, the model automatically eliminates the fully replicated rules from the policy. Note that a policy may contain other types of rule redundancy, described in Section IV-C. Figures 2(a) and 2(b) show an example of an OpenFlow policy and its corresponding PSG model, respectively.

Priority	Match	Actions
2	proto=ssh	drop
1	dstip=1.0.0.2	fwd(2)
1	dstip=1.0.0.3	fwd(3)
0	*	drop

(a) An OpenFlow policy.



(b) A PSG model.

Fig. 2: An OpenFlow policy and its PSG model.

C. SDH Hypervisor

As shown in Fig. 1, each application maintains a network policy specified as an OpenFlow policy. The hypervisor also maintains a policy obtained from a combination of the application policies, called composed policy. As each application submits its policy updates, such as add/delete/insert rules, the hypervisor accordingly updates the composed policy and installs the final updates into the data plane. Note that the details of translating the composed policy into physical policies, including mapping the virtual network to physical network is out of scope of this paper. The network administrator configures the composition between each two application policies using three operators: *sequential*, *parallel*, and *overriding* [12]. In this section, we briefly review these operators.

a) Parallel Composition: The parallel composition between policies P_1 and P_2 applies both policies over all incoming packets and then computes the union of their output packets. For example, one can combine the policies from *Monitoring* and *Routing* applications using the parallel operator. Accordingly, the actions obtained from both policies are applied on any incoming packet.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

b) Sequential Composition: The sequential composition between policies P_1 and P_2 applies P_2 after P_1 over the incoming traffic. To this end, the hypervisor first applies policy P_1 on the traffic and it then applies policy P_2 on the resulted traffic. For example, and administrator can use the sequential composition to combine the policies from *Load Balancer* and *Router* applications. The incoming packets first pass the load balancer policy which might manipulates the packet header. The routing policy is then applied on the new packet headers.

c) Overriding Composition: Such composition of policies P_1 and P_2 applies policy P_2 over the incoming traffic which does not match P_1. The overriding operator can be used to define a default policy.

IV. Anomaly-Free Policy Composition

A. Solution Overview

The main idea behind our policy composition is to model the application policies using PSG and then combine the PSG models to obtain a new PSG which specifies the composed policy. We assume that each application sends a list of OpenFlow rules as *policy updates* to our system. The updates may request to add, delete, or insert rules into the application policy. Our system compiles the updates and generates a new list of rules for the data plane.

Fig. 3 shows our anomaly-free frameworkcontaining two layers: in the upper layer there is a policy manager for each application and in the lower layer there is one policy manager for the composed policy. Each policy manager maintains a PSG to specify its policy. The updates from an application first enter into the corresponding policy manager in which the updates are checked for intra-policy anomalies such as rule redundancy. The updates passed from the anomaly detection are considered for updating the PSG model in the manager which generates updates for the lower layer. The composed policy manager employs the application-layer PSG models to apply a typical policy composition and obtain a list of add/delete rules to update the composed policy. After that, the updates are checked for both intra and inter policy anomalies, the composed PSG is updated accordingly, and finally the updates are sent to the data plane.

It is worth noting that we augmented the application policies to submit updates including rule insertion. This provides an opportunity for an application manager to either insert rules in the middle of the policy or update the priority of rules. This is a common requirement supported in traditional rule-based security tools such as firewalls and VPNs. As one can see in Fig. 3, an application can generate insertion updates and the application policy manager efficiently transforms such updates into several Add/Delete updates. Thus, the composed policy manager only accepts Add/Delete updates. The detailed insertion transformation is described in Section IV-E.

B. Policy Construction

As described, each policy in our solution is represented by a PSG. Thus, the hypervisor maintains one PSG for every application and one for the composed policy. It is clear that the main operations with high time complexity in both policy composition and anomaly detection are set theoretic operations, such as union and intersection, among rule matches. Thus, we employ a multidimensional Patricia trie to maintain the list of peers used in each policy. The multidimensional trie

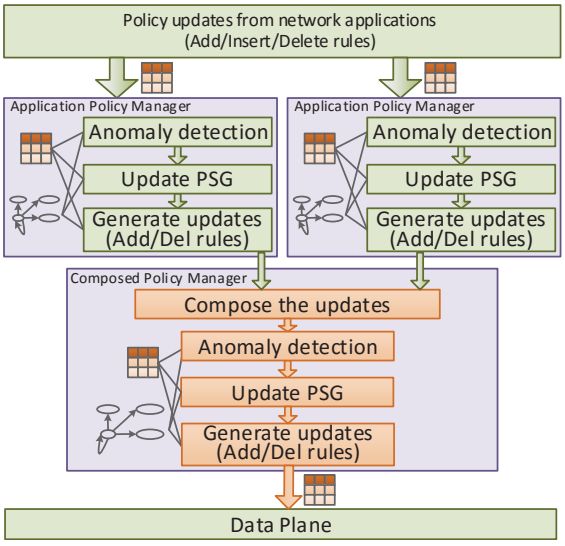


Fig. 3: Our policy composition framework.

is widely used for packet classification [22] and SDNs [23]. We instead use a Patricia trie which is a compressed trie and stores data in every node. Since all the matching fields hold the DisjointOrSubset property (as shown in Proposition 1), we can efficiently store them in the trie. In our data structure, each level in the trie is corresponding to a matching field. Also, each node which contains data, has a link to the root node of a lower-level trie specifying the next matching field. This data structure helps us to obtain the union and intersection using postfix and prefix functions in the tries. Note that each field is represented by a bit stream which help us to make a trie for all values of the field within a policy.

In order to build the PSG model, we employ an incremental method in which for any new rule, the corresponding nodes and edge are created in the model. We also use an adjacency list to represent the PSG model while its nodes are stored in a multidimensional trie.

As described in the previous section, in order to implement the sequential composition, we need the intersection between matching packets after applying the actions of rules. In order to improve the computational complexity of the sequential composition, we maintain another trie which specifies the results of applying all the rules within the policy. Using this idea, we sacrifice the space complexity to reduce the time complexity of the sequential composition to same as the complexity of the parallel composition.

C. Anomaly Detection

Al-Shaer et al. [6] introduced four types of pairwise anomalies among rules in a policy: *Shadowing*, *Correlation*, *Generalization* and *Redundancy*. Basi et al. [24] also classified the anomalies into two categories: *conflict* where a packet is matched with multiple rules with conflicting actions, and *suboptimality* where there is a rule such that its removal has no effect on the policy. We extend these pairwise anomalies to hidden anomalies between a rule and a set of other rules in the policy, called *total anomalies* [9]. Now, we leverage the PSG model to efficiently detect the anomalies in a policy.

We concentrate on redundancy anomalies as the detection algorithms for other anomalies are mostly similar. Moreover, this avoids adding the redundant rules into the policy which

reduces the number updates for the policy. Note that avoiding the redundant rules in the application policies can quadratically decrease both the number of rules in the composed policy and the number of updates installed in the data plane.

Definition 2. ***(Simple Redundancy)*** *An OpenFlow rule is simply redundant in a policy if its rule match is overlapped with an existing rule with a higher priority in the policy. More formally, rule r is simply redundant in policy P if and only if*

$$\exists r_i \in P : r.priority \leq r_i.priority \wedge r.speer \subseteq r_i.speer \wedge r.dpeer \subseteq r_i.dpeer.$$

A naive algorithm for detecting simple redundancy of a new rule is to traverse over all rules with a higher priority than such rule. Thus its complexity would be in $O(n)$, where n in the number of rules in the policy. However, one needs to investigate only rules whose both peers are supersets of the peers of the new rule. We leverage the trie structure to obtains such rules for checking the simple redundancy. Clearly, if the new rule is redundant, it is ignored and the system generates a message to notify the administrator of this redundancy.

Definition 3. ***(Total Redundancy)*** *An OpenFlow rule is totally redundant in a policy if its rule match is overlapped with a set of existing rules with higher priority in the policy. More formally, rule r is totally redundant in policy P if and only if the following formula is a tautology.*

$$r.match.formula \rightarrow \bigvee_{\substack{r_i \in P \\ r.priority \leq r_i.priority}} r_i.match.formula \qquad (2)$$

Similarly to the detection algorithm for the simple redundancy, one can search over all rules with a higher priority than the new rule to construct the propositional formula on the right hand side of the implication in Eq. (2). This equation then is transformed into a propositional logic formula. If the formula obtained from the equation is a *tautology* (a proposition that is always true), the new rule is totally redundant. Now the redundancy detection is transformed to a theorem proving problem in propositional logic. A typical method for theorem proving in propositional logic is using Binary Decision Diagrams (BDDs) [25].

As we have shown, an OpenFlow rule can be encoded as a propositional logic formula using maximum 208 variables. This formula can be represented as a 208-variable BDD. Accordingly, we can build a BDD to represent Eq. 2 for each new rule. After that, the obtained BDD is checked to validate the equation. Although the procedure seems straightforward, the size of the BDD exponentially increases as the number of rules in the policy increases [25]. This is because of many variables used in the decoding of an OpenFlow rule match.

The idea for reducing the size of the BDD obtained from Eq. (2) is to consider only effective rules for transforming the equation to a propositional logic formula. Assume that the hypervisor wants to add a new rule r to policy P. An existing rule $r_i \in P$ is effective for the redundancy detection of rule r if all of the following conditions are held:

- The priority of r_i is greater than or equal to the priority of r, $r_i.priority \geq r.priority$;
- There exists some packets matched by both rules. In other words, the formula $r.match.formula \wedge r_i.match.formula$ is not a contradiction (a proposition that is always false);
- Assume the current formula obtained for the right side of Eq. (2), is denoted by f. Adding rule r_i to f is effective if formula $r_i.match.formula \rightarrow f$ is not a tautology.

It is clear that if an existing rule r_i satisfies the above conditions, it certainly contributes to the right side of Eq. (2). The last condition avoids adding rules which already overlapped with the obtained formula for the right side. By combining the last two conditions, we only check if formula $(r.match.formula \wedge r_i.match.formula) \rightarrow f$ is a tautology. Algorithm 1 shows the procedure for detecting total redundancy of a new rule r with policy P.

Algorithm 1 Total redundancy detection.

```
1:  procedure TOTALREDUNDANCY(PSG P, Rule r)
2:      f ← BDD(False)
3:      for all rule r_i ∈ P do
4:          if r.priority ≤ r_i.priority then
5:              f' ← BDD((r.match ∧ r_i.match) → f)
6:              if f' is not a tautology then
7:                  f ← BDD(f ∨ f')
8:              end if
9:          end if
10:     end for
11:     f ← BDD(r.match → f)
12:     if f is a tautology then
13:         return True
14:     end if
15:     return False
16: end procedure
```

D. Policy Composition

Jin et al. [4] proposed CoVisor, an incremental policy composition algorithm for making small changes to the composed policy every time an application policy changes. Basically, an incremental composition does not need to recompute and reinstall all rules in the composed policy for every single update in an application policy. Instead, it leverages a priority assignment mechanism to generate and install only rules which are related to the update. We employ the calculus proposed in CoVisor for prioritizing the new rules in the composed policy and show how our PSG model improves the efficiency of the policy composition. Note that the overriding composition is implemented using a straightforward algorithm as there is no need to apply any set operators among rule matches. Thus, we take advantage of our PSG model to implement the parallel and sequential compositions.

a) Parallel Composition: In order to combine two policies P_1 and P_2 using the parallel composition, for any two rules $r_1 \in P_1$ and $r_2 \in P_2$ we add a new rule r_k along two original rules r_1 and r_2 to the composed policy. The rule r_k is added into the composed policy if the rule matches of two base rules are not disjoint and r_k is thus obtained as

$$r_k.match = r_1.match \cap r_2.match$$
$$r_k.actions = r_1.actions \cup r_2.actions$$
$$r_k.priority = r_1.priority + r_2.priority$$

Without loss of generality, we assume that a request arrives to add a new rule r_1 to application policy P_1 and the hypervisor wants to obtain all corresponding updates in the composed policy. A naive algorithm is to traverse all rules in P_2 and create all possible rules, such as r_k described above. However, we leverage the trie structure of peers in the PSG model of P_2 to efficiently obtained all rules in P_2 which has intersection with r_1. To this end, we first obtain two sets SP and DP which are overlapping peers with $r_1.speer$ and $r_1.dpeer$, respectively. Then, the edges in PSG from a node in SP to a node in DP represent all the rules intersected with r_1. Note that the updates generated from the policy composition are used to update the PSG model of the composed policy. Clearly, some of the updates may be removed in the anomaly resolution of the PSG model.

b) Sequential Composition: Similar to the parallel composition, we assume that new rule r_1 is added to policy P_1 and the hypervisor can use a similar method to obtain the updates for sequential composition of P_1 and P_2. The only difference is that instead of $r_1.speer$ and $r_1.dpeer$ the hypervisor uses the peers $r_1.speer'$ and $r_1.dpeer'$ which respectively correspond to $r_1.speer$ and $r_1.dpeer$ after applying the actions of r_1.

In the case that a new rule r_2 is added to policy P_2, the hypervisor must obtain the rules in P_1 which their matches after applying their actions have intersection with $r_2.match$. As discussed in the model construction phase, the hypervisor maintains two PSG models for each policy, one for the original policy and another for modeling the peers after applying the actions. We employ the second PSG model of P_1 to obtain a subset of rules in P_1 must be sequentially combined with r_2.

E. Insertion Transformation

The methods described in the previous sections implement both add and delete rules to/from an application policy and consequently to/from the composed policy in the hypervisor. Since the flow tables in an OpenFlow switch support redundant priority for the flow entries [19], adding a rule is implemented without changing the priority of other rules in the policy. However, many rule-based security systems allow the administrator to manipulate the priority of existing rules as well as inserting a rule in a specific position with the policy.

For example, assume that the administrator of the policy shown in Fig. 2 wants to forward all the packets coming from source IP 2.0.0.5 to port number 5 except for the *http* packets which must be dropped. Now, the administrator can simply insert a rule "`2;srcip=2.0.0.5;fwd(5)`" between the first two rules in the policy. To this end, the first rule must be shifted in order to make a free space for the new rule. This is because there is a rule with an identical priority and with an overlapped rule match with the new rule in the policy.

It is worth noting that the inserting operation is a very beneficial feature in rule-based applications as the administrator can locally decide about the position of a new rule without checking whole of the rules in the policy. A naive method to implement the insert operation is to first shift all the higher priority rules in order to make free space for the new rule. After that, the insert operation is transformed into an add operation. Accordingly, if there are n rules with a higher priority than the new rule, this method first removes these n rules, adds them while increments their priority, and finally add the new rule in the free space. Thus, the naive method needs $2n + 1 = O(n)$ update operations. Note that such number of updates generated for updating an application layer policy can lead to an update with a quadratic length, $O(n^2)$ for either a parallel or sequentially composed policy.

It is clear that the main objective of the rule priorities is to prioritize the overlapping rules in the match process as the OpenFlow policy uses a *first match* mechanism. In other words, such priority can be disregarded when the match rules are disjoint. By leveraging this fact we formulate our approach based on two propositions: 1) we can limit the propagation of any priority update to only the rules intersected with the updated rule; and 2) if the policy contains a free space for the inserted rule, we need no rule shifting. Note that we augment the application policy updates allows an administrator to insert rules. The hypervisor uses our approach to transform the insert operation into a list of add/delete updates for such policy. It then sends these updates to the policy composition module to generate corresponding updates for the composed policy.

Algorithm 2 shows our approach for transforming an insert update requested for an application policy into a list of add/delete updates. The method proposed for model construction is used for applying the add/delete updates in the application policy. Moreover, the composed policy is updated for the updates based on the composition algorithms described in the previous section.

Algorithm 2 Transforming an insert rule into add/delete rules.

Input: P: a PSG, r: a new rule to be inserted into P
Output: U: A set of add/delete updates

```
1:  procedure TRANSFORMINSERT(PSG P, Rule r)
2:      U ← {Add(r)}
3:      if ∄r' ∈ P, r'.priority = r.priority then
4:          return U
5:      end if
6:      for all rule r' ∈ P do
7:          if r.priority ≤ r'.priority then
8:              f ← BDD(r.match.formula ∧ r'.match.formula)
9:              if f is not a contradiction then
10:                 U ← U ∪ {Del(r')}
11:                 r'' ← Clone(r')
12:                 r''.priority ← r''.priority + 1
13:                 U ← U ∪ {Add(r'')}
14:             end if
15:         end if
16:     end for
17:     return U
18: end procedure
```

F. PSG to Flow Table

Although we propose an incremental method for updating the flow tables in the data plane, a hypervisor might regenerate and reinstall the rules into the data plane. Thus, we need to efficiently translate the PSG model into forwarding tables that represent packet processing in the switches.

As we described, a path from root to a leaf node in the multi-dimensional trie represents an existing peer in the policy. Such a leaf node has an edge to another leaf node for each

rule in which the rule participates as a source peer. Thus, in order to generate a set of forwarding rules from a PSG model, one can use a DFS-like algorithm for traversing the whole of the trie and generate the rules for each source peer. Note that the PSG model contains the priority of the rules and there is no need to sort the rules before deploying them into the data plane. Moreover, the forwarding rules generated using this approach from a PSG model is anomaly-free as we resolves the anomalies during the model updating process.

V. Experimental Evaluation

A. Experimental Environment

We implemented a prototype of our method on the top of CoVisor[1] which itself is a part of the OpenVirteX network hypervisor [26]. We selected this platform for two reasons. First, this allows us to conduct a comparative evaluation against CoVisor, a recently proposed policy composition method [4]. Second, our policy composition works independently and is compatible with the built-in virtualizations in both OpenVirteX and CoVisor. We used the `PatriciaTrie` API[2] to implement the trie for each matching field. We extended the library to also give us a sorted map of objects that are not disjoint by a key. This helps to efficiently detect both simple and total anomalies. We also used the `JavaBDD` library[3] to manipulate BDDs needed for our anomaly detection mechanism.

We assume that the applications' administrators generate their policy update requests as a list of add/insert/delete rules. Thus, in all experiments we evaluate the efficiency of the hypervisor for generating and installing the corresponding updates for the composed policy based on four types of metrics: 1) **update length** which is the average number of updated flows in the composed policy per each update in the application policy; 2) **total length** which is the composed policy length after compiling the updates in the application policies; 3) **compile time** which is the average elapsed time needed to compile one update from an application policy into the composed policy; and 4) **total time** which is the total elapsed time needed to compile one update and install the corresponding generated updates into the data plane.

We conducted experiments to evaluate the performance of the proposed policy compositions and insertion transformation. In all these experiments, the application policies are obtained based on the filter sets generated by ClassBench [27] and we randomly create the rule updates. All the experiments were conducted on an iMac PC with 2.00GHz Intel Core 2 Duo processor and 4GB RAM running Ubuntu 12.04 LTS. The program code was written in Java with JDK 1.7.

B. Policy Composition

In order to conduct a fair comparative evaluation of the policy composition, we follow the evaluation method proposed in [4]. Accordingly, in this section we consider a network with two applications where the hypervisor composed their policy using either a parallel, sequential or overriding composition operator. To measure the efficiency metrics, we first install the base policy of both applications into the hypervisor, and then measure the performance of policy updating by adding 10 uniformly randomly selected rules into both applications. The process is repeated 100 times and then results are averaged. We also vary the number of rules in the base policies for both application to investigate the performance of the composition algorithms in different policy lengths. For all experiments, we measure the performance metrics for both the proposed method, called *PSG* and *CoVisor* proposed in [4].

Fig. 4 shows the performance results of both approaches for a parallel composition between two applications such as Monitor and Router. As one can see in Fig. 4(a), the rule update overhead in CoVisor linearly increases as the number of rules increases in the base policies while the overhead is approximately steady for PSG. Moreover, the number of flows generated per each update in PSG is always less than CoVisor. This can be explained by the fact that the number of anomalous rules increases in the larger base policies and consequently this dramatically raises the number of unnecessary rules generated in the composed policy. A similar trend can be observed in the composed policy length shown in Fig. 4(b). As we expected, the total number of flows in the composed policy generated by PSG is considerably less than the policy generated by CoVisor. This is because of many unnecessary flows in both base and composed policies were not treated in CoVisor.

Figures 4(c) and 4(d) show the compile time and total time of composition per rule update, respectively. The results show that the policy compilation using PSG takes longer than CoVisor. This is due to additional processing for anomaly detection and resolution based on algorithms described in Section IV-C. However, PSG is significantly faster than CoVisor according to the total time of policy composition shown in Fig. 4(d). This can be explained by the fact that PSG sacrifices a few milliseconds to resolve the anomalies and consequently it considerably reduces the number of updates needed to be installed in the data plane. Since the hypervisor needs to communicate with the switches to install the updates, reducing the update length through eliminating the unnecessary rules leads to a significant efficiency in the network.

In the second scenario, we assume a sequential composition between two applications such as a Firewall and a Router. Fig. 5 shows the performance results of this composition. Again, PSG is several order of magnitude efficient than CoVisor in terms of number of rules generated per each update as well as the length of composed policy. Although that PSG is slightly slower than CoVisor in compilation of a sequential composition, it is significantly faster than CoVisor according to the total composition time as it considerably filters the anomalous rules.

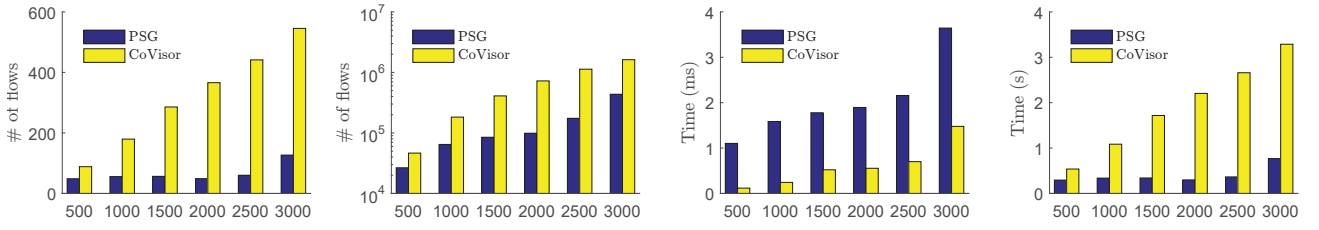
Fig. 6 illustrates the performance results of an overriding composition between two application policies. As one can see in Fig. 6(a), CoVisor always generates one update rule for the composed policy while the average number of update rules generated by PSG is less than one. Note that the maximum updates per one rule added in overriding composition is one rule in the composed policy as we have employed an incremental mechanism. Thus, since PSG is an anomaly-free composition method, it eliminates the anomalous rules and consequently its average update length is less than one.

[1] http://covisor.cs.princeton.edu/

[2] http://docs.oracle.com/javase/6/docs/technotes/guides/collections/

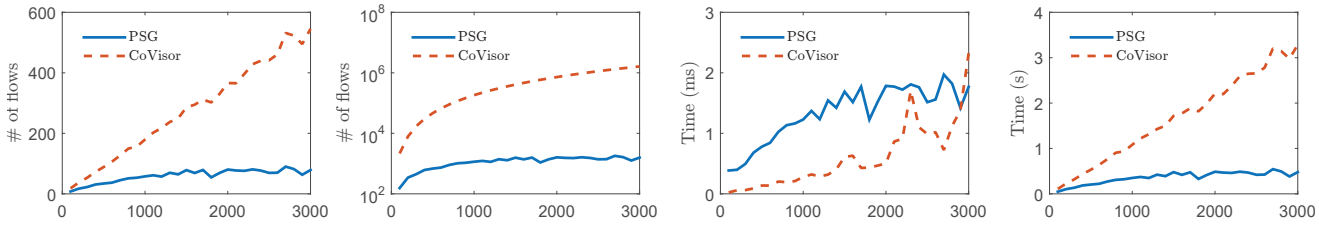
[3] http://javabdd.sourceforge.net/

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



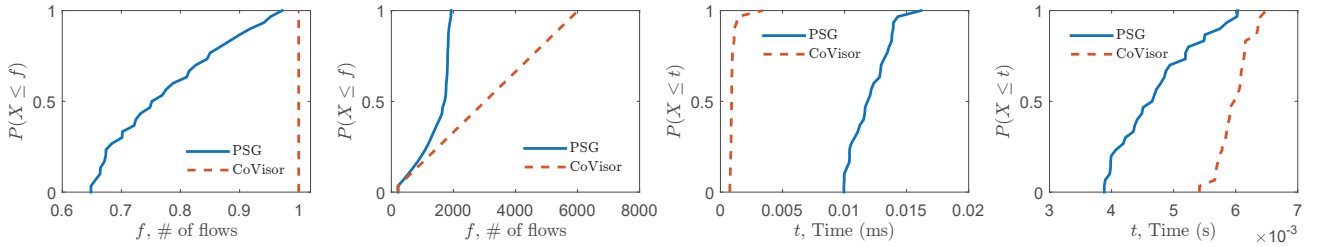
(a) Update length. (b) Total length. (c) Compile time. (d) Total time.

Fig. 4: Update performance for parallel composition.



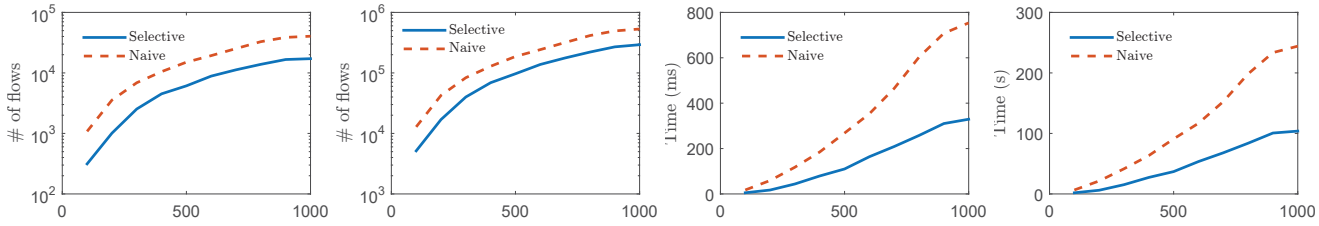
(a) Update length. (b) Total length. (c) Compile time. (d) Total time.

Fig. 5: Update performance for Sequential composition.



(a) Update length. (b) Total length. (c) Compile time. (d) Total time.

Fig. 6: Update performance for Override composition.



(a) Update length. (b) Total length. (c) Compile time. (d) Total time.

Fig. 7: Update performance for rule insertion into parallel composition.

C. Rule Insertion

In this section, we evaluate the performance of our method for translating insert operations for both single and composed policies. In the first experiment, we consider a single application policy and then randomly select a portion (2%, 5%, or fixed update length of 10) of rules and generate updates for inserting these rules. The base policy in this experiment is generated from firewall filter set by ClassBench [27]. We measure the update size for our insert translation, called *Selective* and the naive approach, called *Naive* for each process. As described in Section IV-E, an insertion request will be transformed into a set of add and delete rule requests. Thus, we consider the total number of these requests as the update length of insertion transformation. We also use the PSG composition approach for both cases which help us to eliminate anomalous rules. The process is repeated 100 times and then results are averaged.

Table I shows the average update length generated for insertions into a single policy. The results shows that the update generated by the Selective approach is often more than 10 order magnitude smaller than that of the Naive method.

In the second experiment, we evaluate the performance of rule insertion into a combined policy. To this end, we use the settings of the experiment for parallel composition in the previous section. Then, the updates is randomly generated by 10 rule insertion into the application policies. Fig. 7 shows the performance results for insertion into a parallel composition policy. As one can see in this figure, Selective

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

TABLE I: Average rule updates for inserting into a policy.

Base	2%		5%		Fixed=10	
Rules #	Naive	Selective	Naive	Selective	Naive	Selective
100	60.66	2.93	63.29	3.72	63.68	4.07
300	230.55	16.38	232.61	16.38	235.15	16.43
500	362.79	30.14	366.32	31.72	362.79	30.14
700	460.32	39.95	460.98	39.52	461.99	40.71
900	531.75	53.25	522.89	48.36	526.28	48.95
1100	614.15	60.87	559.30	52.08	585.98	62.40
1300	650.30	62.29	595.64	54.09	625.94	66.76
1500	695.12	64.10	610.86	57.22	657.91	67.46

usually generates updates with a length of around half of the length of updates generated by Naive. Moreover, the total length of composed policies generated by these two approach are approximately in the same order.

Figures 7(d) and 7(d) illustrate that both the compilation and total time of Naive considerably increase with the policy size, because larger policies force more updates including both deleting and adding rules. Since these updates need to be applied into the composition procedure, Naive is becoming much slower. On the other hand, Selective has a slight growth in both the compilation and total time, because it generates much less updates in transforming the insertion for the composition.

VI. Conclusions

In this paper, we proposed a novel framework for policy composition in a SDN hypervisor which considers both intra and inter anomaly detection and resolution. Moreover, we augmented the framework to efficiently transform priority change updates, such as rule insertion, in the application-level policies into the composed policy. We plan to extend the PSG model for bitmask feature proposed in the recent OpenFlow version. Moreover, supporting a network abstraction in PSG can help us to detect and resolve run-time anomalies in the network policies. We also plan to take advantage of multi-table support in OpenFlow to improve the performance of the policy composition.

Acknowledgment

This work was funded in part by the Google Faculty Award 2015 (Jha & Pagnucco).

References

[1] O. N. Foundation, "Software-defined networking: The new norm for networks," ONF White Paper, Tech. Rep., April 2012.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, 2008.

[3] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software-defined networks," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13, 2013, pp. 1–14.

[4] X. Jin, J. Gossels, J. Rexford, and D. Walker, "CoVisor: A compositional hypervisor for software-defined networks," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15, 2015, pp. 87–101.

[5] X. Wen, C. Diao, X. Zhao, Y. Chen, L. E. Li, B. Yang, and K. Bu, "Compiling minimum incremental update for modular SDN languages," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14, 2014, pp. 193–198.

[6] E. S. Al-Shaer and H. H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*.

[7] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra, "FIREMAN: A toolkit for firewall modeling and analysis," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, 2006, pp. 199–213.

[8] H. Hu, G.-J. Ahn, and K. Kulkarni, "Detecting and resolving firewall policy anomalies," *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 3, pp. 318–331, May 2012.

[9] M. Rezvani and R. Aryan, "Analyzing and resolving anomalies in firewall security policies based on propositional logic," in *IEEE 13th International Multi Topic Conference, INMIC*, 2009.

[10] R. Jalili and M. Rezvani, "Specification and verification of security policies in firewalls," in *Proceedings of the First EurAsian Conference on Information and Communication Technology*, 2002, pp. 154–163.

[11] P. Adao, C. Bozzato, G. D. Rossi, R. Focardi, and F. Luccio, "Mignis: A semantic based tool for firewall configuration," in *IEEE Computer Security Foundations Workshop*, July 2014, pp. 351 – 365.

[12] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," *SIGPLAN Not.*, vol. 46, no. 9, pp. 279–291, Sep. 2011.

[13] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker, "NetKAT: Semantic foundations for networks," *SIGPLAN Not.*, vol. 49, no. 1, pp. 113–126, Jan. 2014.

[14] W. Han, H. Hu, and G.-J. Ahn, "LPM: Layered policy management for software-defined networks," in *Data and Applications Security and Privacy XXVIII*, ser. LNCS, 2014, vol. 8566, pp. 356–363.

[15] A. Dwaraki, S. Seetharaman, S. Natarajan, and T. Wolf, "GitFlow: Flow revision management for software-defined networks," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15, 2015.

[16] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular composable security services for software-defined networks," in *Proceedings of the ISOC Network and Distributed System Security Symposium*, February 2013.

[17] C. Prakash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "PGA: Using graphs to express and automatically reconcile network policies," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15, 2015, pp. 29–42.

[18] S. Smolka, S. Eliopoulos, N. Foster, and A. Guha, "A fast compiler for NetKAT," in *Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming*, 2015, pp. 328–341.

[19] "Openflow switch specification version 1.5.0," *https://www.opennetworking.org/*, 2014.

[20] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated openflow infrastructures," in *Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration*, ser. SafeConfig '10. New York, NY, USA: ACM, 2010, pp. 37–44.

[21] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'13, 2013, pp. 99–112.

[22] G. Varghese, *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices*. Elsevier/Morgan Kaufmann, 2005.

[23] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, 2012, pp. 49–54.

[24] C. Basile, A. Cappadonia, and A. Lioy, "Network-level access control policy analysis and transformation," *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 985–998, Aug. 2012.

[25] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, vol. 35, no. 8, pp. 677–691, Aug. 1986.

[26] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "OpenVirteX: Make your virtual SDNs programmable," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. ACM, 2014, pp. 25–30.

[27] D. E. Taylor and J. S. Turner, "ClassBench: A packet classification benchmark," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 499–511, 2007.

Resilience of Interdependent Communication and Power Distribution Networks against Cascading Failures

Wei Koong Chai, Vaios Kyritsis, Konstantinos V. Katsaros, George Pavlou
Department of Electronic and Electrical Engineering
University College London
London, WC1E 6BT, United Kingdom
Email: {w.chai, vaios.kyritsis.13, k.katsaros, g.pavlou}@ucl.ac.uk

***Abstract*—The operations of many modern cyber-physical systems, such as *smart grids*, are based on increasingly interdependent networks. The impact of cascading failures on such networks has recently received significant attention due to the corresponding effect of these failures on the society. In this paper, we conduct an empirical study on the robustness of interdependent systems formed by the coupling of power grids and communication networks by putting real distribution power grids to the test. We focus on the assessment of the robustness of a large set of medium-voltage (MV) distribution grids, currently operating live in the Netherlands, against cascading failures initiated by different types of faults / attacks. We consider both unintentional random failures and malicious targeted attacks which gradually degrade the capability of the entire system and we evaluate their respective consequences. Our study shows that current MV grids are highly vulnerable to such cascades of failures. Furthermore, we discover that a small-world communication network structure lends itself to the robustness of the interdependent system. Also interestingly enough, we discover that the formation of hub hierarchies, which is known to enhance independent network robustness, actually has detrimental effects against cascading failures. Based on real MV grid topologies, our study yields realistic insights which can be employed as a set of practical guidelines for distribution system operators (DSOs) to design effective grid protection schemes.**

I. Introduction

Our modern society is increasingly reliant on networks for various aspects of life, ranging from basic needs (*e.g.*, energy supplies) to those contributing to better standard of living (*e.g.*, transportation, information systems). The integration of these increasingly intelligent and critical infrastructure networks, in turn, has also made the various originally separate networks dependent on each other (*e.g.*, a cyber-network overlaying a physical-network). While the strong coupling of networks enhances their functionalities, it also significantly increases the vulnerability of the system as a whole [1], [2]. This is because failures in one network may cascade to the other and vice versa, resulting in an iterative failure process. Hence, robustness of such interdependent network systems (sometimes known as network of networks (NoN)) has recently received much attention (see for example [3]).

In this paper, we focus our study on the resilience of interdependent networks consisting of an electric power grid and a communication network against such cascading failures since the energy and telecommunication sectors are found to be the main sectors initiating such cascade of failures [4], [5]. The symbiotic relationship between the two networks is a result of the grid requiring the service from the communication network for monitoring, control and management operations, while the communication network depends on the grid for electricity supply. A real-world example demonstrating such interdependency and the corresponding vulnerability is the national blackout in Italy in September 2003 [1]. This interdependency is expected to increase with the advent of smart grids introducing bi-directional communication patterns among multiple entities.

Our work focuses on the medium-voltage (MV) distribution grid domain, which is recently undergoing transformative changes due to the advent of smart grid applications, but have yet to receive the same level of attention as its high-voltage (HV) counterpart (*e.g.*, as highlighted in [6], [7]). The introduction of multiple dynamic active components, *e.g.*, distributed (renewable) energy resources (DERs) such as solar/wind farms and electric vehicles (EVs), at the distribution level poses new significant challenges to the system stability especially on protection and reliability of the grid. The traditional assumptions of distribution networks being mostly passive and static no longer apply as they evolve towards the so-called Active Distribution Networks (ADNs)[1] where increased fine-grained observability of the grid power conditions, faster response and enhanced protection are needed to manage the increased volatility of the system in a timely fashion. The operation and, more importantly, the protection of distribution grids are thus increasingly reliant on a robust and efficient communication infrastructure that must provide seamless and timely communication service, such that full observability of power conditions is maintained at all times [8]. In fact, to cater for the challenges of next generation smart grids, there are already work in the literature to apply the latest information-centric networking paradigm for smart grid applications (*e.g.*, [9], [10], [11]).

However, the communication network landscape in the MV domain is far from clear [12]. The distribution grid (i.e., the MV domain) covers smaller geographical areas compared to HV grids, as well as areas of different nature *i.e.*, rural

[1]ADNs are distribution networks that have systems in place to control a combination of DERs (*e.g.*, generators, loads and storage). DSOs have the possibility of managing electricity flows via a flexible network topology. DERs take some responsibility for system support, depending on a suitable regulatory environment and connection agreement.

ISBN 978-3-901882-83-8

but also denser (sub-)urban areas. As a result, the adoption of technologies employed in the HV domain to our case is not straightforward, mainly due to the associated deployment costs [12]. Furthermore, distribution system operators (DSOs) have multiple options for the communication network, such as engaging the service of public Internet service provider(s) (ISPs), deploying their own private communication infrastructure (*e.g.*, investing in fibre optic and/or exploiting existing powerline communication (PLC) technologies, such as that proposed in [12]) or adopting a hybrid solution, using both the private and public options above. In view of this still evolving communication environment, we employ widely used network topology models with different characteristics, as the communication network component of the interdependent system, to gain insights on the robustness of the overall NoN.

Our work aims to close the gap in the study on the vulnerability of interdependent systems against cascading failures, which hitherto mainly focused on purely theoretical analysis (cf. Section II for model descriptions). For instance, focusing on a special case where the system consists of two totally identical networks dependent on each other, the authors in [13] studied the system using algebraic connectivity of an interdependent system as the robustness indicator. In [1], the robustness of the interdependent network system is studied assuming totally uncorrelated networks using percolation theory. Since it is known that such uncorrelated networks do not exist in the real world and almost always the interdependent networks do not share common topologies, we put real distribution grids to the test *i.e.*, we use a large set of real MV grid networks currently operating live in the Netherlands by a major Dutch DSO, engaging thus in an extensive empirical study (cf. Section III) to quantitatively gain insights into the system behaviour. Furthermore, we focus our study on the MV domain which is increasingly becoming more dynamic and thus, more prone to such cascading failures. While the communication network landscape in the MV domain is still shaping, our findings provide practical guidelines on the desirable topological characteristics that can enhance system robustness when designing / deploying the communication infrastructure over the distribution grid. Moreover, our study broadens the set of failure types encountered in the considered NoN. Namely, Hines *et al.* reported that the majority of cascading failures in power grids are results of natural disasters, but also highlighted the recent increase of cyber-attacks initiating blackouts via hacking of the communication network [14]. Our study covers a spectrum of different failure types, ranging from unintentional faults to malicious attacks, each characterized by specific node removal pattern (cf. Sections III-C1 and III-C2). Additionally, instead of considering one single failure that initiates a cascade of failures, we consider a more general context where multiple cascading failures occur. We discuss our observations and insights, providing a better understanding of the system behavior under failures and thus, facilitating the design of effective protection schemes against different types of cascading failures (cf. Section IV). We summarize and conclude our findings in Section V.

II. Cascading Failures in Interdependent Systems

We consider an interdependent system with two undirected graphs, $G^{sg} = (V^{sg}, E^{sg})$ and $G^{com} = (V^{com}, E^{com})$, representing the (smart) power grid and communication network respectively. Let N^{net} be the network size where $net \in \{sg, com\}$. Then, $V^{net} = v_1^{net}, \ldots, v_{N^{net}}^{net}$ and $E^{net} = e_1^{net}, \ldots, e_{M^{net}}^{net}$, where M^{net} is the number of edges / links in the corresponding network. Further, let A^{net} denote the $N^{net} \times N^{net}$ adjacency matrix with the elements, $A_{i,j}^{net} = 1$ if there exists a link between nodes i and j and 0 otherwise. The interdependency of the two graphs is represented by an $N^{sg} \times N^{com}$ matrix A^{dep} with $A_{v_i^{sg}, v_j^{com}}^{dep} = 1$ if there exists a link between v_i^{sg} and v_j^{com} and 0 otherwise.

A. Interdependent System Model

For this work, without loss of generality, we establish the baseline interdependent system as follows:

- Both networks are equal in size, $N^{sg} = N^{com} = N$, but they do not necessarily possess the same topologies (as opposed to [13] where $A^{sg} = A^{com}$ is assumed).
- There is 1-to-1 dependency between nodes in G^{sg} and G^{com} (*i.e.*, there are N interdependency links in the system). This can correspond to cases where each S-SS is equipped with a communication network node for the support of monitoring and control applications, *e.g.*, [8].
- The dependency is bi-directional (*i.e.*, mutually dependent). A node failure in G^{sg} will result in the failure of the corresponding dependent node in G^{com} and vice versa.
- Drawing on the observations of real-world interdependent systems reported in [15], we follow the *positive* degree correlation method and create dependency between nodes with similar level of degrees (*i.e.*, nodes with high degree (nodes having many immediate neighbors) in one network are coupled with nodes having high degrees in the other network and vice versa [16])[2].

Real-world interdependent systems may not always have such "balanced" interdependencies. Nevertheless, it is straightforward to accommodate unbalanced cases such as n-to-m node inter-network connections and non-symmetric dependencies (*e.g.*, in [17]) in our methodology (cf. Section III). In this case, we note system robustness may be enhanced with higher number of interdependency links (*i.e.*, a node may only fail when all of its counterpart nodes in the other network fails). However, as studied in [18], the cost to achieve the added robustness must be carefully considered.

B. Cascading Failure Model

In this work, we follow the cascading failure model described in [1] which has since been widely used in the literature as the basis of several studies (*e.g.*, [2], [19], [20], [21]). In this model, a cascading failure begins with the failure (*i.e.*,

[2]We have also experimented with *random* and *negative* degree correlations for creating interdependency between the two networks but insignificant divergences are observed.

[3]We leave out specific practical details of power grid in the example such as the switching of P-SS for power source or possibility of islanding operations.

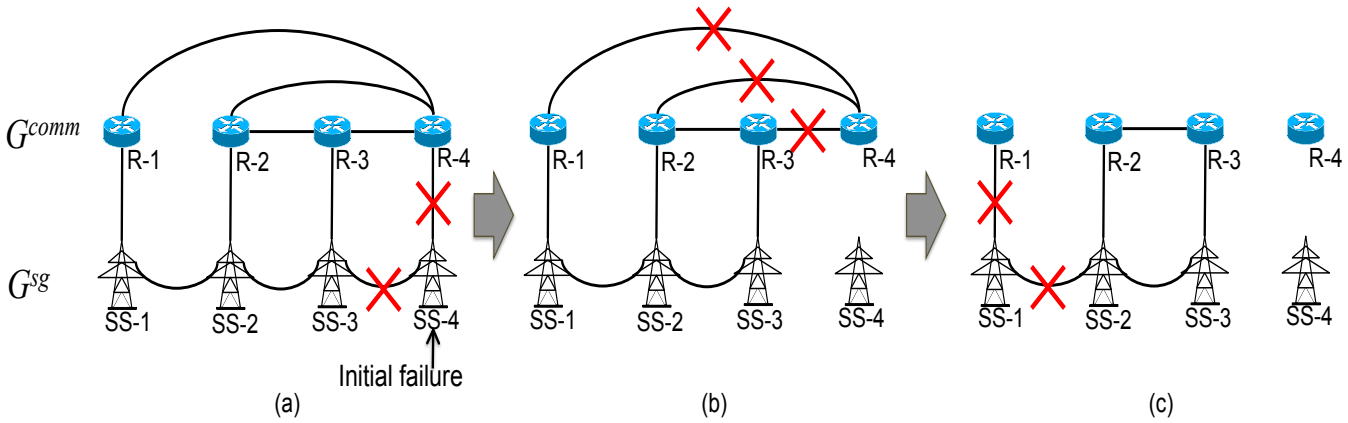


Fig. 1: (Color Online) An illustration of a cascading failure suffered by an interdependent system of size $N^{sg} = N^{com} = N = 4$ triggered by a failure at node SS-4[3].

removal) of a fraction, $1 - p$ of nodes from network **A**. All the links connecting to these failed nodes in network **A** will thus be down. Further, the nodes in network **B** depending on these nodes will also fail. Any link connected to these failed nodes in network **B** will also be removed. This process may fragment both networks and form different network components since the two networks are differently connected. The nodes belonging to the largest mutually connected component (*i.e.*, the giant component) retain their functionality while the smaller components fail.

Figure 1 illustrates a simple example of a cascading failure triggered by the failure of one node (*i.e.*, $N(1-p) = 1$) in an interdependent system consisting of $N = 4$ nodes in each network. The initial failure occurs at node SS-4 in G^{sg} causing its removal along with the link to node SS-3 and the interdependent link to node R-4 in G^{com}. The loss of the interdependent link causes the failure of node R-4 and of all the links connected to it. This causes G^{com} to fragment whereby only the giant component retains its functionality i.e., nodes R-2, R-3 and the link connecting them. The fragmented smaller clusters fail. This cascades back to G^{sg} causing the failure of node SS-1 and of all links connected to it.

III. Methodology

A. Network Models

In practice, the complete knowledge of how the separate networks are dependent on each other may not always be possible since multiple stakeholders are often involved. In our case, when DSOs rely on communication infrastructure providers, the exact communication network information (*e.g.*, the network topology) is confidential[4]. In our study, we create the interdependent systems using real data for G^{sg}. However, as the communication network environment in the MV domain still evolves, we resort to widely accepted theoretical graph models for G^{com}. Within this context, our objective is to investigate the key structural properties that improve the resilience of the considered interdependent networks, deriving practical guidelines for the design of communication networks for the MV domain.

Smart grid network, $\mathbf{G^{sg}}$ – we use real data extracted from 16 MV distribution grids, covering an area of approximately $350km^2$ in central eastern Netherlands. They include a total of 16 primary sub-stations (P-SSes) and 1,857 secondary sub-stations (S-SSes). The grid topologies resemble that of tree structures rooted at P-SS(es), which perform(s) the high-to-medium voltage transformation. Each tree branch emanating from the P-SS corresponds to a distinct feeder. Table I shows the basic topological characteristics of these grids[5] and Fig. 2 shows the aggregated degree distribution of all the sub-stations (SSes) across the entire set of MV grids. Almost 90% of SSes have low number of neighbors (*i.e.*, degree of one or two) while approximately 20% of SSes are leaf nodes (*i.e.*, nodes connected to only one other node).

Communication network, $\mathbf{G^{com}}$ – we use three main theoretical graph models widely used in the literature to study network robustness (*e.g.*, [22]) to gain insights into the graph properties that would deter/promote cascading failures.

- Erdős-Rényi (ER) model – Given N, a link randomly connects a pair of nodes with probability p_r independent of all other links. In our experiments, we use $p_r = \frac{lnN}{N}$ which is the sharp threshold of connectedness to ensure connected graphs while at the same time sufficiently small to avoid a highly meshed topology [23]. ER graphs are characterized by a short average path length and low clustering coefficient, since a consequence of pure random edge allocation is that the degree distribution converges to a Poisson distribution. The simplicity of the model has lent itself to many theoretical studies on network resilience (*e.g.*, [1]). In our case, the ER model can be considered as an "unplanned" network layer that is resulted from gradual ad hoc deployment of network nodes to incrementally support the new requirements from the grid over time.
- Small world (SW) model – We construct SW graphs following the Watts and Strogatz model [24], with

[4]This non-disclosure is mutual as DSOs also do not offer information regarding their own power grid.

[5]We "anonymize" the grids by removing location/power related information.

rewiring probability, $\gamma = 0.1$ and mean degree, $\bar{k} = 0.05 \times N$. A low γ value is used to avoid creating SW graphs that closely resemble ER graphs since when $\gamma = 1$, the resultant graph's average path length converges to that of a random graph (*i.e.*, $ln(N)/ln(\bar{k})$). In addition, when $\gamma << 1$, SW graphs also form local clusters (*i.e.*, having high clustering coefficient) as opposed to graphs such as lattices which exhibit the opposite characteristic ("big world" graphs). Moreover, since our work focuses on MV grids which usually do not span over large spatial proximity, there is high probability that the corresponding G^{com} will exhibit SW properties.

- Scale-free (SF) model – In our study, SF graphs are constructed based on the Barabási-Albert (BA) model [25], using a 3-node seed graph. In this model, each new node is connected to an existing node with a probability proportional to the existing nodes' degree (*i.e.*, the more neighbors a node has, the more likely it attracts a new node to attach to it). Owing to this preferential attachment process, SF graphs result in power law degree distribution. This property has been observed in many real-world networks (*e.g.*, [26]) which results in the forming of hubs within the graph. Since there is no prevailing communication network design for the support of smart grid applications in the MV domain, we also investigate the effect of the interdependent system when coupled with SF graph topologies.

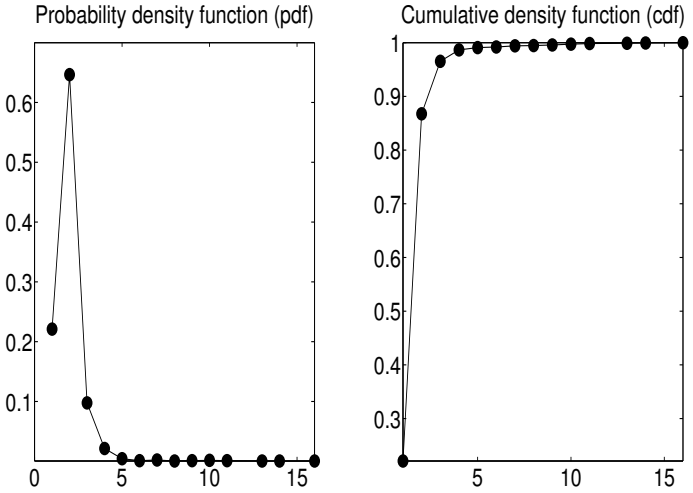


Fig. 2: Aggregated degree distribution of sub-stations in the dataset showing majority of sub-stations have low connectivity.

B. Metrics

To evaluate the impact of cascading failures, we follow the literature to use the size of the giant component which measures the level of connectivity of a network as an evaluation metric (*e.g.*, [1], [2], [16]). In practice, the functional component also depends on the actual power source (*i.e.*, P-SS), the re-configurability of the grid (*e.g.*, locations of breakers)

[6]Link density = $\frac{M^{net}}{N^{net}(N^{net}-1)/2}$.

TABLE I: Properties of real MV grid topologies of a large European DNO.[6]

Grid	N^{sg}	M^{sg}	Mean degree	Clustering coefficient	Link density	Mean path length
Area 1	126	126	2.000	0.0039	0.0160	8.3194
Area 2	83	82	1.9759	0.0000	0.0241	6.5601
Area 3	191	192	2.0105	0.0035	0.0106	9.6177
Area 4	23	22	1.9130	0.0000	0.0870	5.5652
Area 5	178	177	1.9888	0.0000	0.0112	10.0176
Area 6	29	28	1.9310	0.0000	0.0690	8.0296
Area 7	51	51	2.0000	0.0000	0.0400	7.1529
Area 8	102	101	1.9804	0.0000	0.0196	9.3345
Area 9	99	98	1.9798	0.0000	0.0202	8.4684
Area 10	209	210	2.0096	0.0000	0.0097	11.6043
Area 11	47	47	2.0000	0.0156	0.0435	8.1082
Area 12	294	298	2.0272	0.0024	0.0069	11.8471
Area 13	85	84	1.9765	0.0000	0.0235	8.1815
Area 14	42	41	1.9524	0.0000	0.0476	4.7433
Area 15	146	147	2.0137	0.0000	0.0139	9.7879
Area 16	168	169	2.0119	0.0000	0.0120	10.5974

as well as the ability to perform islanding operations. In this sense, the nodes in the system are not homogeneous and have different resiliency in reality. For instance, nodes located near breakers may be more easily switched to another power source and thus, in certain cases, becoming less vulnerable. Due to the fact that these specific cases are dependent on the distinct operations of the network, we take a topological approach to draw more general insights that should be applicable to different interdependent systems.

In addition, to understand how the communication functionality degrades, we measure the communication efficiency of the G^{com}, η following [27]:

$$\eta = \frac{\sum_{1 \le i < j \le N} 1/\sigma_{i,j}}{\binom{N}{2}} \quad (1)$$

where $\sigma_{i,j}$ is the shortest path length (in hopcount) between node i and j. It measures how fast information spreads in a network. A fully mesh network has $\eta = 1$ since all nodes are reachable in one hop (*i.e.*, all node pairs have 1-hop distance). This metric is especially relevant to time-critical smart grid applications such as synchrophasor-based monitoring where the measurements taken at geographically distributed locations are synchronized and must reach the phasor data concentrators (PDCs) within very stringent time window [8], [12].

C. Failure Models

Our investigation of the considered NoN's resilience properties is based on a sequential process where we remove one node after another, with each removal triggering a cascading failure each time. This results in a series of cascading failures. Throughout this process, we track the gradual degradation of the system functionalities in terms of both the above-mentioned metrics. The order in which nodes are removed from the system depends on the failure type. In the following, we describe the different types of failures considered in this work.

1) Unintentional Random Failures: Unintentional failures include those caused by equipment failures, natural disasters (*e.g.*, earthquake, tsunami, *etc.*) or simply accidental human errors (*e.g.*, misconfigurations). For such failures, the sites where they take place are usually non-determinable or forecasted. Such failures are modeled via random node removals. We use random point (RP) node removals for failures caused by equipment failures or human errors and random area (RA) node removals for failures caused by natural disasters that usually spread in a specific geographical area. Namely,

- Random Point (RP) failure – Given $1-p$, $N(1-p)$ nodes are randomly selected for removal.
- Random Area (RA) failure – Given $1-p$, start by removing a random node in the graph and then proceed to remove a random neighbor of the removed node that has survived the resulting cascading failure due to the removal of the initial node. Repeat the process of removing random neighbors of removed nodes until $N(1-p)$ nodes have been removed.

2) Malicious Targeted Attacks: Malicious attacks aim to maximize damage to the interdependent system by targetting parts of the system believed to be vulnerable. Such attacks may come in physical form, via equipment tampering or electronically via intentional misconfigurations or spreading of computer viruses. To conduct such an attack, the perpetrator must possess some prior intelligence regarding the targeted system such as knowledge on the topologies and their interdependencies. Logically, with the intention to cause maximum damage, the attacker will attack nodes deemed to be most important to the system operation. We assume that the attacker ranks the nodes based on their importance in descending order and attacks the system in that order.

To compute this ranking, we consider four centrality measures, widely used when studying network robustness [28], [29]:

- Node degree (DC) – relates node importance with the number of immediate neighbors *i.e.*, local connectivity.
- Betweenness (BC) – measures the involvement of a node in the set of shortest paths of all node pairs in the network
- Closeness (CC) – measures the distance of a node to all other nodes in a connected network
- Eigenvector (EC) – relates node importance to the importance of its neighbors

Each centrality measure above deduce importance of nodes based on different factor: DC – connectivity, BC – path, CC – distance and EC – spectral structure of the topology. In [29], an in-depth comparative assessment of these centrality measures in the context of communication networks is conducted.

We extend the centrality concepts to account for the added importance of each node with regards to its counterpart in the other network. Specifically, for each node, v, we compute the mean value of the normalized node centrality within its own network and those node(s) in the other network that have a connection to it. Table II shows the original centrality definitions and Eq. 2 gives the extended definitions.

TABLE II: Centrality indices used for targeted attacks.

Centrality Index	Definition
Degree (DC)	$c^{DC}(v) = \frac{deg(v)}{N^\alpha - 1}$
Betweenness (BC)	$c^{BC}(v) = \frac{2}{(N^\alpha-1)(N^\alpha-2)} \sum_{i\neq v\neq j \in V^\alpha} \frac{\sigma_{i,j}(v)}{\sigma_{i,j}}$
Closeness (CC)	$c^{CC}(v) = \frac{N^\alpha - 1}{\sum_{j\in V^\alpha, i\neq j} \sigma_{i,j}}$
Eigenvector (EC)	$c^{EC}(v) = \frac{1}{\lambda} \sum_{j\in V^\alpha} A^\alpha_{v,j} \times c^{EC}(j)$
N: graph size, α, β: indicates which network (either *sg* or *com*), $deg(v)$: number of neighbors of node v within its own network, λ: eigenvalue, $\sigma_{i,j}$: shortest path length from i to j, $\sigma_{i,j}(v)$: shortest path length via v	

For each centrality index, $x \in \{DC, BC, CC, EC\}$, we extend them to the coupled network system as follows:

$$c^x_{ext}(v) = \frac{c^x(v) + \sum_{j\in V^\beta} A^{dep}_{v,j} \times c^x(j)}{1 + \sum_{j\in V^\beta} A^{dep}_{v,j}}. \quad (2)$$

The above equation assumes the importance of a node is proportionally increased based on the (total) importance of the node(s) depending on it (*i.e.*, additive effect). Note that Eq. 2 is universally applicable to unbalanced interdependent systems (cf. as discussed in Section II-A).

IV. Vulnerability Analysis

We conducted an extensive simulation study across 16 real MV grids coupled with three types of communication network models (*i.e.*, $\{ER, SW, SF\}$) against six types of failure trigger patterns (*i.e.*, $\{RA, RP, DC, BC, CC, EC\}$) (cf. Section III). For each simulation setup, we obtained 95% confidence intervals for all metrics. For each repeat simulation run, we regenerated a new G^{com} since reusing the same one results in exact same node ranking for targeted attacks. Due to the large number of possible scenarios, we present selected but representative results and discuss our observations and findings.

A. Impact of Random Cascading Failures

We first show in Fig. 3 representative results of MV grid coupling with different G^{com} models. From all sets of results, we observe that MV grids coupled with SF networks are the least robust against both types of random cascading failures ($SF \prec ER \prec SW$[7]) with the size of the giant component rapidly decreasing. This corroborates with [1] where interdependent SF graphs with different power-law degree distributions are found to be more vulnerable to interdependent ER graphs. For the RA case, the degradation of the system is close for MV grids dependent on ER and SW networks. The distinction becomes less clear for small distribution grids (see insets in Fig. 3). The observed order of robustness, $SF \prec ER \prec SW$, is also observed in [22]

[7]To simplify discussion, we use $X \prec Y$ ($X \succ Y$) to indicate that X is less (more) robust against cascading failures than Y within the considered setup.

which considers failures in single layer network models with no interdependency.

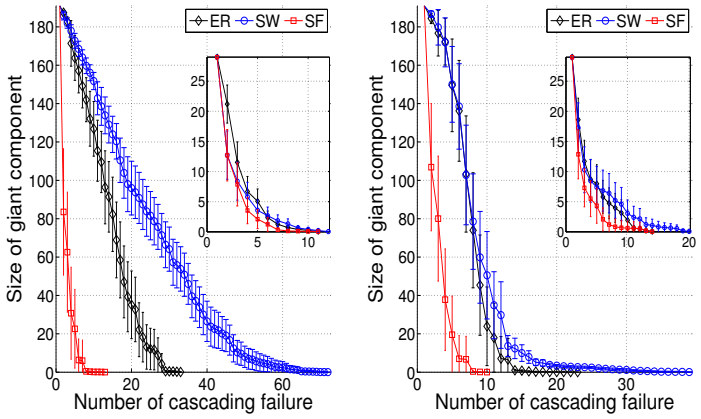


Fig. 3: (Color Online) The impact of cascading failures with RP (left) and RA (right) failures for an MV grid with $N > 30$. (Insets: Results for an MV grid with $N < 30$.)

We next show in Fig. 4 how the network degrades in terms of the communication efficiency at G^{com} and the size of the giant component at G^{sg}. Both metrics suffer similar deterioration for all cases. RP causes relatively less severe functionality degradation (RA $\succ$ RP). However, when $1-p$ increases, we observe consistently that at one point the reverse trend (RA $\prec$ RP) becomes true. This behavior is also observed in small MV grids with $N < 30$ (see insets of Fig. 4). This indicates that when the spreading effect of cascading failures is limited (*i.e.*, when the network system is small), randomized failure points cause more detrimental impact than the gradual failure of immediate neighbors of failed nodes.

B. Impact of Targeted Cascading Failures

To get a better understanding of the different types of targeted cascading failures, as expressed by our extended centrality (Eq. 2), we first investigate the extend to which different centrality measures result in attacks on different nodes. To this end, we use Spearman coefficients, as a full rank correlation proxy, and the percentage of top-10% node overlap, as a high rank-correlation proxy. Tables III and IV give a sample Spearman coefficient and top-10% overlap respectively. The Spearman coefficients across all the MV grids show that the centrality pairs have *low* correlations. In fact, extended EC often *negatively* correlates with others. The observation is consistent for high rank nodes (top-10% overlap) where the overlap tends to be low. As such, the actual targeted nodes (both the set of nodes and the order) are different when based on the different extended centrality rankings.

Interestingly, despite this, our results show that their impact to the system is rather similar. Figure 5 shows the impact of the different attacks on different network interdependencies where we observe overlapping curves. This behavior is consistent across all the MV grids, suggesting that the MV grid topology structures are especially vulnerable to cascading failures in general, regardless of the points of attack. This indicates that simple protection schemes protecting nodes with high centrality are not sufficient to defend MV grids against cascading failures. This observation agrees with the theoretical findings in [2]. Our observations here generalize [2]'s conclusion to include different types of attacks (*i.e.*, not only for degree-based attacks).

TABLE III: Sample Spearman coefficient for MV grid coupled with different G^{com} models.

	DC	BC	CC	EC	G^{com}
DC	1 1 1				ER SW SF
BC	0.5120 0.4636 0.8485	1 1 1			-//-
CC	0.7100 0.5258 0.3308	0.5155 0.5234 0.3851	1 1 1		-//-
EC	-0.7854 0.1728 -0.0622	-0.4488 -0.3482 -0.1734	-0.9162 -0.4642 0.2818	1 1 1	-//-

TABLE IV: Sample top-10% overlap (%) for MV grid coupled with different G^{com} models.

	DC	BC	CC	EC	G^{com}
DC	1 1 1				ER SW SF
BC	0.3158 0.4737 0.4211	1 1 1			-//-
CC	0.5263 0.5790 0.2105	0.5790 0.6316 0.4211	1 1 1		-//-
EC	0 0.3158 0.4211	0 0 0.1579	0 0.0526 0.2105	1 1 1	-//-

For all the three coupling cases, {ER, SW, SF}, we found that the different targeted attacks are very effective (*i.e.*, the size of the giant component and communication efficiency decrease rapidly after only approximately 10% of nodes removed). This is attributed to two factors: (1) the MV grids have (near-)zero clustering coefficient (see Table I) and (2) high number of nodes with degree $= 2$ ($\approx 65\%$ of total nodes, see Fig. 2). The co-existence of these properties results in high probability of network fragmentation as there is very good chance that a failure involves a "bridge" node that singularly connects the grid network. As only the giant component survives, such fragmentation rapidly disintegrates the system. Moreover, we note that the MV grids also have low path diversity and exhibit relatively long mean path lengths which further increase the importance of the "bridge" nodes in maintaining connectivity.

For all types of targeted attacks, we observe the robustness follows $SF \prec ER \prec SW$ order which is consistent with that observed for random failure cases. Therefore, for the case of MV grids, they are most resilient against cascading failures when dependent on a communication network exhibiting small-world properties such as low average path lengths and

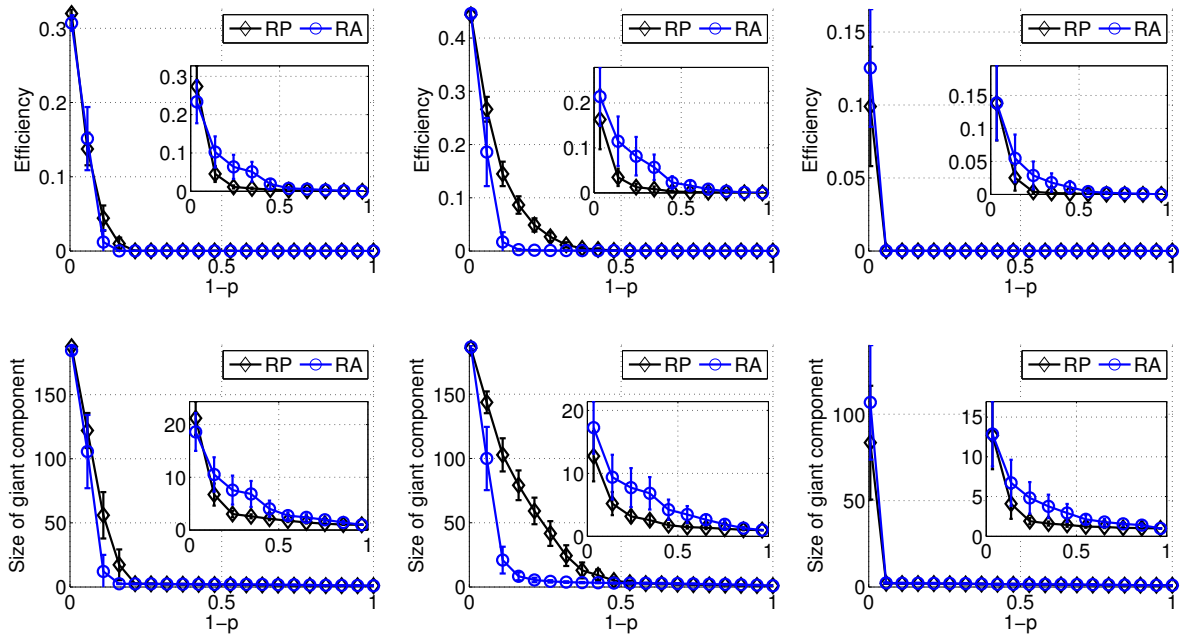


Fig. 4: (Color Online) The impact of random cascading failures on the communication efficiency (top row) and the size of giant component for an MV grid (bottom row) when $1-p$ fraction of nodes are removed based on RP and RA strategies for an MV grid with $N > 30$ (Inset: grid with $N < 30$) coupled with ER (left column), SW (center column) and SF (right column) graph.

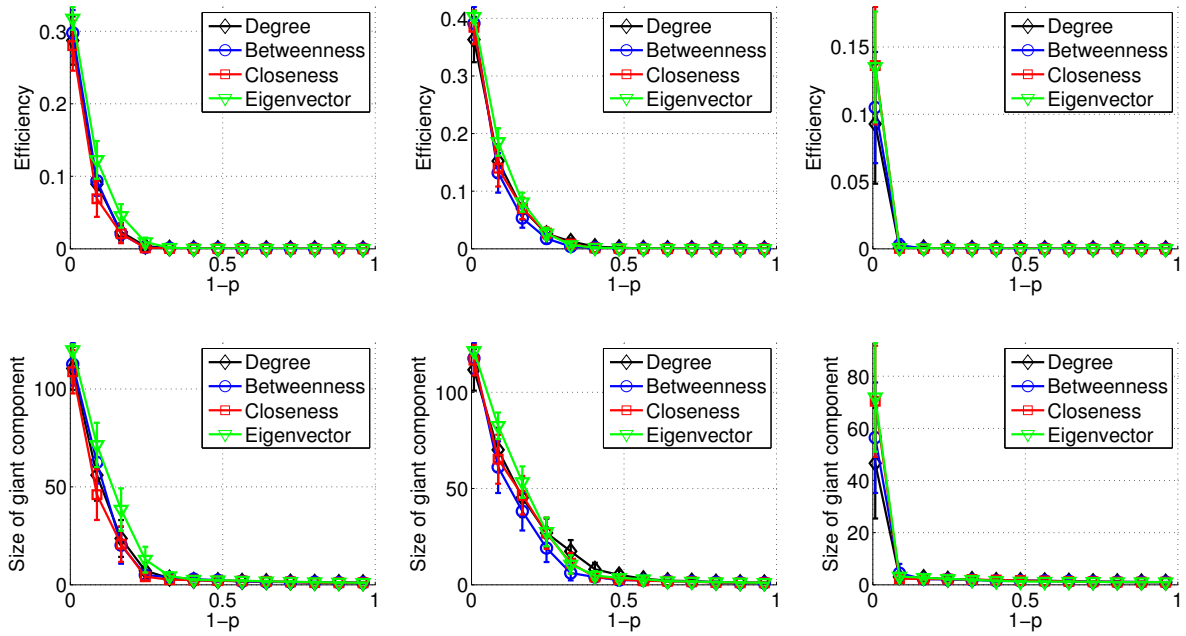


Fig. 5: (Color Online) The impact of cascading failures on the communication efficiency, η (top row) and the size of giant component (bottom row) when $1-p$ fraction of nodes are removed based on different targeted attacks for an MV grid coupled with $N > 30$ interdependent on ER (left column), SW (center column) and SF (right column) graph.

high clustering coefficient. Conversely, MV grids coupled with SF networks are always the least robust against cascading failures. On one hand, this corroborates with past analysis for targeted attacks (*e.g.*, [30]) since simultaneous removal of top well connected hubs rapidly fragments the network. On the other hand, this finding is also counter-intuitive as SF networks are known to be robust to random removals [25] due to the fact that (1) most nodes have small degree (non-hub) and (2) major hubs are usually connected to other smaller hubs; forming a hierarchy of hubs that resists network fragmentation. Tracking the system following our sequential cascading failure simulations, we found that such vulnerability

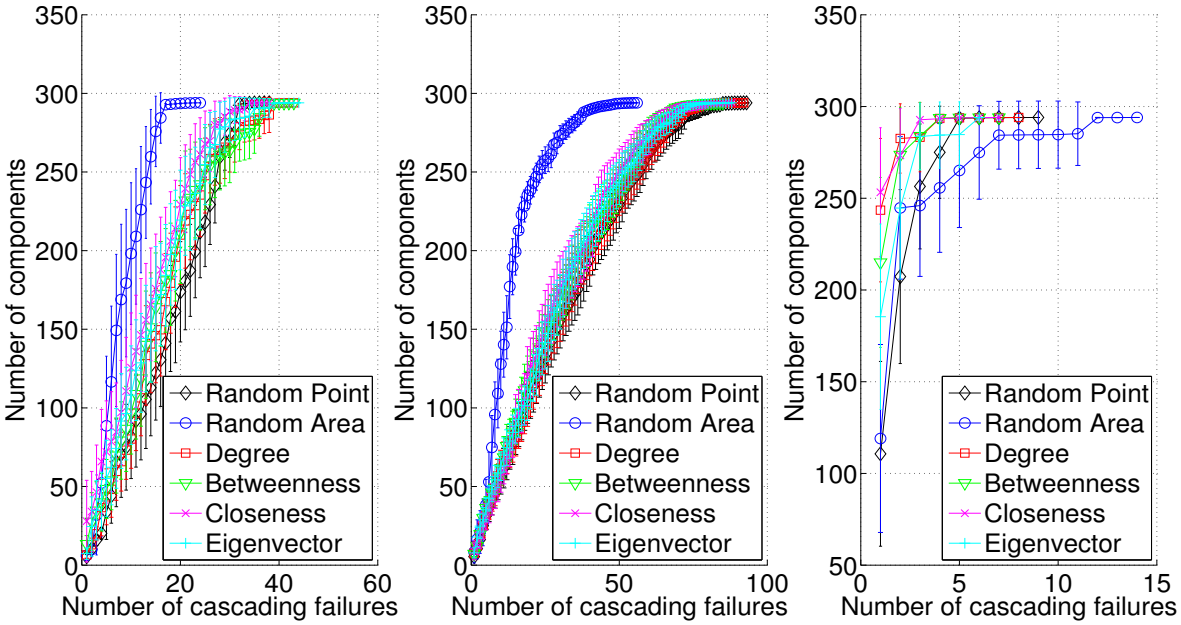


Fig. 6: (Color Online) Effect of cascading failures on the number of resulting components for MV grid coupled with ER (left), SW (center) and SF (right) networks.

is due to the fact that cascading failures have a spreading effect which in most cases, involves the removal of multiple hubs in a single cascade. Hence, the existence of multiple hubs surrounding a hub that "protects" the network from losing its connectedness turns out to be the very reason why coupling with SF networks is especially vulnerable, as cascading failures have high probability of removing multiple hubs in the near vicinity simultaneously.

For each type of theoretical graph model, we further experimented with varying clustering coefficient and link density. We found no direct relationship between clustering coefficient and the impact of cascading failures (*i.e.*, the size of giant component / communication efficiency) – high clustering coefficient does not necessarily provide better resilience. However, we find that, for relatively sparse networks (*i.e.*, at low link density region), link density can be a good relative robustness indicator with better system robustness found in systems with higher link density. Nevertheless, this is not to be used singularly as a determinant of system robustness as the robustness still depends on the exact degree distribution of the networks. For instance, an MV grid coupled with an SW graph with lower link density may still be more robust than a system coupled with an SF graph with high link density.

Next, we investigate the extreme case where we fail the system gradually until all nodes fail. For this, we relax the assumption that only the giant component retains functionality but allow any component of size > 1 to be functional. This allows us to gain insights into the change (if any) in the disruptive power of individual cascading failures when the networks are gradually disconnected. We show sample results with $N = 191$ in Fig. 6. Systems coupled with SF networks remain to be the most vulnerable ones. The system is completely disintegrated after only approximately $10 \sim 15$ cascading failures for SF networks while interdependent systems that couple with ER and SW networks require approximately 40 and 90 cascading failures respectively. While most failure types cause similar level of damage, RA failures seem to be most effective when MV grids are dependent on ER and SW networks. When MV grids couple with SF networks, RA failure is the least effective (on average) but the confidence interval indicates that the results differ significantly compared against others.

Finally, we show in Fig. 7 the average number of cascading failures required to complete the disintegration of the system (*i.e.*, all nodes disconnected) for the entire set of real MV grids from our dataset. We observe that MV grids coupled with SF networks are so vulnerable that an increase in N does not result in increasing number of cascading failures required. On the other hand, we see the gradual increase of the number of cascading failures required for MV grids coupled with SW networks when N increases, suggesting that small-world properties are beneficial to protect an interdependent system against cascading failures.

V. Summary and Conclusions

In this paper, we study the impact of cascading failures on an interdependent system consisting of a communication network and an MV distribution power grid, using real grid networks that are currently operating in the Netherlands. We evaluate the effect of such iterative failures on the MV grids coupled with different types of communication networks (including random, small world and scale-free network models) and types of failures (both unintentional and intentional failures). Our study shows that MV grids are extremely vulnerable to cascading failures, a finding of particular importance when considering the advent of the smart grid with the increasing interdependency of the grid and supporting communication network. The tree-like structure of MV grids, featuring very

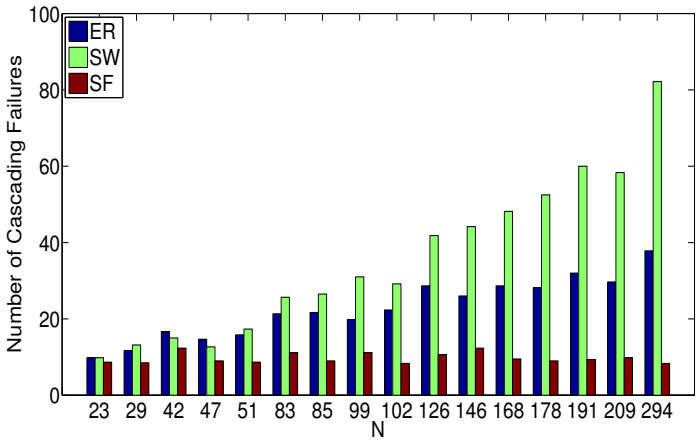


Fig. 7: (Color Online) Number of cascading failures to fully disintegrate the interdependent system.

low clustering coefficient, high mean path lengths and mean node degree close to two, is the main contributing factor to this, since a high number of "bridge" nodes increases the probability of fragmentation of the MV grids. The interdependent system as a whole is almost equally susceptible to catastrophic damage regardless of the nature of failures. Simple protection schemes focusing on protecting specific "important" nodes (*e.g.*, high centrality nodes) will not be effective against cascading failures. Furthermore, broad degree distribution of the communication network topologies, known to strengthen the resilience of network against single non-cascading failures, is found to have the reverse effect on interdependent systems. Specifically, the existence of multiple hubs (as in SF graphs) is detrimental to the system against cascading failures. Coupling with SW networks result in the most robust system, implying small-world properties are beneficial for the robustness of interdependent systems. Finally, we found that higher link density in sparse networks (*i.e.*, at low density region) provides better robustness for the same type of network model.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Commission's Seventh Framework Programme FP7-ICT-2011-8 under grant agreement no 318708 (C-DAX) and the CHIST-ERA / EPSRC UK project CONCERT (grant no. EP/L018535/1). The authors alone are responsible for the content of this paper.

REFERENCES

[1] S. V. Buldyrev, *et. al.*, "Catastrophic cascade of failures in interdependent networks," Nature, vol. 464, no. 7291, pp. 1025-1028, 2010.

[2] X. Huang, *et. al.*, "Robustness of interdependent networks under targeted attack," Physical Review E83, 065101 (R) 2011.

[3] G. D'Agostino and A. Scala, Eds., "Networks of Networks: The Last Frontier of Complexity," New York, NY, USA: Springer, 2014, (DOI) 10.1007/978-3-319-03518-5.

[4] E. Luiijf, *et. al.*, "Empirical findings on critical infrastructure dependencies in Europe," in Critical Information Infrastructure Security, Lecture Notes in Computer Science, Berlin, Germany, Springer, 2009.

[5] E. Zio and G. Sansavini, "Modeling interdependent network systems for identifying cascade-safe operating margins," IEEE Trans. Reliability, vol. 60, no. 1, March 2011.

[6] G. A. Pagani and M. Aiello, "Towards decentralization: A topological investigation of the medium and low voltage grids," IEEE Trans. Smart Grid, vol. 2, no. 3, pp. 538-547, Sept. 2011.

[7] G. A. Pagani and M. Aiello, "The power grid as a complex network: A survey," Physica A 392 (2013) 2688-2700.

[8] W. K. Chai, *et. al.*, "An Information-centric Communication Infrastructure for Real-time State Estimation of Active Distribution Networks," IEEE Trans. Smart Grid, vol. 6, no. 4, pp. 2134-2146, July 2015.

[9] W. K. Chai, *et. al.*, "Enabling Smart Grid Applications with Information-centric Networking," Proc. 2nd ACM Conference on Information-Centric Networking (ICN 2015), San Francisco, USA, Sep. 30 - Oct. 2, 2015.

[10] K. V. Katsaros, *et. al.*, "Supporting Smart Electric Vehicle Charging with Information-Centric Networking," Int'l Workshop on Quality, Reliability, and Security in Information-Centric Networking, Greece, 2014

[11] K. V. Katsaros, *et. al.*, "Information-centric Networking for Machine-to-Machine Data Delivery - A Case Study in Smart Grid Applications," IEEE Networks Magazine, vol. 28, no. 3, pp. 58-64, May-June 2014.

[12] K. V. Katsaros, B. Yang, W. K. Chai and G. Pavlou, "Low Latency Communication Infrastructure for Synchrophasor Applications in Distribution Networks," Proc. SmartGridComm 2014, Venice, Italy, Nov 2014.

[13] J. Martín-Hernández, H. Wang, P. Van Mieghem and G. D'Agostino, "Algebraic Connectivity of Interdependent Networks", Physica A, Statistical Mechanics and its Applications, no. C, vol. 404, pp 92-105, 2014.

[14] P. Hines, K. Balasubramaniam and E. C. Sanchez, "Cascading failures in power grids," IEEE Potentials, vol. 28, no. 5, pp. 24-30, 2009.

[15] V. Rosato, *et. al.*, "Modelling interdependent infrastructures using interacting dynamical models," International Journal of Critical Infrastructures, 4(1/2):63-79, 2008

[16] T. N. Dinh, *et. al.*, "On new approaches of assessing network vulnerability: hardness and approximation," *IEEE/ACM* Trans. Netw., vol. 20, no. 2, pp. 609-619, April 2012

[17] G. Fu, *et. al.*, "Interdependent networks: vulnerability analysis and strategies to limit cascading failure," European Physical Journal B, 87:148, July 2014.

[18] Z. Huang, C. Wang, M. Stojmenovic and A. Nayak, "Balancing system survivability and cost of smart grid via modeling cascading failures," IEEE Transactions on Emerging Topics in Computing, June 2013.

[19] J. Guo, *et. al.*, "Networks formed from interdependent networks," Nature Physics, vol. 8, Jan. 2012, pp. 40-48.

[20] D. T. Nguyen, *et. al.*, "Detecting critical nodes in interdependent power networks for vulnerability assessment," IEEE Trans. Smart Grid, vol. 4, no. 1, pp. 151-159, March 2013.

[21] M. Parandehgheibi, E. Modiano and D. Hay "Mitigating Cascading Failures in Interdependent Power Grids and Communication Networks," in Proc. SmartGridComm 2014, Venice, Italy, Nov 2014.

[22] S. Trajanovski, J. Martín-Hernández, W. Winterbach and P. Van Mieghem, "Robustness envelopes of network," Journal of Complex Networks (2013) 1, pp. 44-62.

[23] P. Erdős, A. Rényi, "On the evolution of random graphs", Mathematical Institute of the Hungarian Academy of Sciences, pp. 17-61, 1960.

[24] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks", Nature 393 (6684); pp. 440-442, 1998.

[25] A. L. Barabasi and R. Albert, "Emergence of scaling in random networks," Science, vol. 286, no. 5439, pp. 509-512, Oct. 1999.

[26] G. Siganos, *et. al.*, "Power laws and the AS-level internet topology," *IEEE/ACM* Trans. Netw. vol. 11, no. 4, pp. 514-524, Aug. 2003.

[27] V. Latora and M. Marchiori, "Efficient behavior of small-world networks," Phys. Rev. Lett., vol. 87, no. 19, 5 Nov 2001.

[28] S. Wasserman and K. Faust, "Social network analysis: Methods and Applications," Cambridge: Cambridge University Press, 1994.

[29] G. Nomikos, P. Pantazopoulos, M. Karaliopoulos and I. Stavrakakis, "Comparative assessment of centrality indices and implications on the vulnerability of ISP networks," 26th International Teletraffic Congress (ITC), 2014.

[30] R. Cohen, K. Erez, D. ben-Avraham and S. Havlin, "Breakdown of the Internet under intentional attack," Phys. Rev. Lett. 86: 36825, 2001.

Connectivity-aware Virtual Network Embedding

Nashid Shahriar*, Reaz Ahmed*, Shihabur Rahman Chowdhury*, Md. Mashrur Alam Khan*, Raouf Boutaba*, Jeebak Mitra†, and Feng Zeng†
*David R. Cheriton School of Computer Science, University of Waterloo
{nshahria | r5ahmed | sr2chowdhury | mmalamkh | rboutaba}@uwaterloo.ca
†Huawei Technologies
{jeebak.mitra | zengfeng137140}@huawei.com

***Abstract*—The problem of ensuring virtual network (VN) connectivity in presence of multiple link failures in the substrate network (SN) is not well investigated in Network Virtualization (NV) literature. We name this problem as Co*nnectivity-aware* Vi*rtual* N*etwork* E*mbedding* (*CoViNE*). Solving *CoViNE* will enable a VN operator to perform failure recovery without depending on the SN provider, similar to the IP restoration mechanisms in IP-over-WDM networks. There are two steps in solving *CoViNE*: i) finding the virtual links that should be embedded disjointly, and ii) finding a substrate resource efficient embedding that ensures the virtual link disjointness constraint. We present two solutions to the *CoViNE* problem. The first solution uses a heuristic to compute the disjointness constraint, while an optimization model is used for VN embedding. The second solution, in contrast, uses heuristic for both the steps, and thus can solve larger instances of the problem. We compare our solutions with a cut set based approach that ensures VN connectivity for a single substrate link failure. Evaluation results show that our heuristics allocate $\sim$15% extra resources on average compared to the cut set based optimal solution, and executes two to three orders of magnitude faster on the same problem instances.**

I. Introduction

Perceived as a key enabling technology for the future Internet, Network Virtualization (NV) offers efficient resource sharing by embedding multiple Virtual Networks (VNs) on a single Substrate Network (SN). One of the major challenges in NV is VN embedding (VNE) [1], *i.e.*, to find a mapping of the virtual nodes and links onto substrate nodes and paths, respectively without violating physical resource constraints.

Substrate resources may fail. Surviving failures is of paramount importance, since a single failure in SN may result into multiple failures in the embedded VNs. Finding a VN embedding that can survive failures in SN is known as the Survivable Virtual Network Embedding (SVNE) problem [2]. Majority of the works on SVNE focus on link failures, as they occur more frequently than node failures [3]. SVNE approaches, in general, allocate redundant bandwidth for each (or selected) virtual link(s), either proactively while computing the embedding or reactively after a failure occurs [4].

In this paper, we focus on a different form of survivability than traditional SVNE, which we call **Co***nnectivty-aware* **Vi***rtual* **N***etwork* **E***mbedding* (*CoViNE*). Our goal is to find a VN embedding that can ensure connectivity in a VN topology in presence of multiple substrate link failures. In contrast, SVNE approaches focus on guaranteeing virtual link demand in presence of failure(s). SVNE approaches assume that SN-providers hide physical failures by over-provisioning a fraction of each virtual link's bandwidth, which in turn incurs additional cost to VN-operators. In contrast, we ensure VN connectivity, which is a weaker form of survivability incurring lesser resource overhead and reduced cost of leasing. We intend to empower a VN-operator to handle link failures according to its internal policy, e.g., customer priority. A VN-operator can plan and over-provision different amounts of bandwidth in each virtual link to handle failures according to its needs, instead of blindly relying on the SN-provider that over provisions fixed bandwidth for each virtual link as in SVNE approaches. Connectivity aware embedding will enable a VN-operator to reroute traffic on failed virtual links, which can be done using any IP link restoration protocol.

Although our focus is NV, the *CoViNE* problem is equally applicable in IP-over-Wavelength-division multiplexing (WDM) domain. The problem of ensuring IP layer connectivity in presence of a single WDM link failure is known as *link survivable mapping*. Two variations of the problem have been studied in IP-over-WDM literature [5]: i) *weakly link survivable mapping* (WLSM) ensures IP-layer connectivity; ii) *strong link survivable* mapping guarantees both connectivity and bandwidth of the failed IP link(s) in presence of a single WDM link failure. WLSM, which considers single link failure, is merely a special case of *CoViNE*.

Despite being neglected in the literature, we focus on multiple (more specifically up to double) link failures, since it is not a rare event in large transport networks. First, repairing a failed link (e.g., due to fiber cut) can take long time [3]. Chances of a second link failure is not negligible given the high Mean-Time-to-Repair (MTTR). Second, some inter-datacenter links destined to different places may be physically routed together for some distance, and a backhaul failure may cause multiple physical links to fail [6]. It can be derived from the statistics in [7] that $\sim$12% of the failures in inter-datacenter transport networks are double link failures in SN.

There are two conditions for surviving multiple (say, k) substrate link failures: i) the VN topology must be $k+1$ edge connected, and ii) the embedding algorithm must ensure at least $k+1$ edge-disjoint paths in SN between every pair of virtual nodes. The first condition can be satisfied by augmenting the VN with new links [8]. A naive way to satisfy the second condition is to embed all the virtual links of a $k+1$ edge connected VN onto disjoint paths in the SN. However, this is an NP-complete problem [9], and imposes an

ISBN 978-3-901882-83-8

unsatisfiable number of disjointness constraints. Existing cut set based approaches suffer from poor scalability [10], [11]. Furthermore, most of the heuristic schemes either focus on single link failure [12], [13], or fail to deal with arbitrary VN topologies [14]. The approach in [8] proposes a generalized solution for multiple link failures. This solution requires a large number of virtual links to be embedded disjointly, hence, is not resource efficient. Therefore, we propose novel solutions that embed arbitrary VN topologies with near-optimal disjointness constraints to survive in presence of multiple link failures. Our solutions augment a VN with minimal number of virtual links while preserving the topological structure of the VN. The major contributions of this paper are:

1) We explore an alternate survivability model, *CoViNE*, requiring significantly less backup resources than traditional survivability approaches in SVNE literature.
2) We present two generalized solutions to the *CoViNE* problem dealing with multiple substrate link failures. The first solution builds upon heuristic for augmenting the VN and computing the virtual links that should be embedded disjointly, and an optimization model for VN embedding adhering to the disjointness requirement. The second solution, in contrast, uses heuristic for both the steps and thus can solve larger instances of the problem.
3) Through extensive simulations, we evaluate the optimality and the time-complexity of the proposed solutions. In addition, we show how *CoViNE* can reduce the impact of failure in single and double link failure scenarios.

The rest of this paper is organized as follows. We present the related literature in Section II. In Section III, we present the system model and problem statement. A theoretical foundation of *CoViNE* is laid in Section IV. A heuristic algorithm for computing disjointness constraint is presented in Section V. An optimization model and a heuristic algorithm for VN embedding are presented in Section VI and Section VII, respectively. We present the simulation setup and evaluation results in Section VIII. Finally, we conclude with future research directions in Section IX.

II. Related Works

A number of approaches exist in the literature for survivable VN Embedding. However, these approaches mostly focus on ensuring the same end-to-end QoS guarantee after single SLink failure [2], [15], [16], [17]. A number of research works in IP-over-WDM literature focus on ensuring connectivity of IP links under WDM link failures [10], [13], [14]. In this section, we briefly describe the most prominent approaches in literature, and contrast them with our solutions for *CoViNE*.

Modiano *et al.* [10] presented an ILP formulation for survivable VLink routing on WDM SN in presence of single SLink failure. Their formulation explores exponential number of cut sets in the VN and routes all the VLinks of a cut set on disjoint WDM paths. Todimala *et al.* [11] improved the ILP formulation by identifying polynomial number of primary cuts in a VN. The authors in [18] extended the Max-flow min-cut theorem for multi-layer networks and proposed approximation algorithms for VLink routing, while maximizing the minimum cross layer cut. A major drawback of these approaches is that they do not scale well with network size, because of the inherent complexity of LP-solvers.

Several heuristic based approaches have been proposed for survivable VN embedding in large networks. Kurant *et al.* [14] proposed SMART, a framework for finding survivable mapping of a VN by repeatedly picking cycles of a VN and finding survivable mappings for the cycles. SMART can ensure connectivity under double SLink failures for VNs having a few special structures, hence, has limited applicability. An extension to SMART has been proposed by [12] that exploits the duality between circuits and cuts in the VN. Zhou *et al.* [13] proposed an algorithm that identifies a set of spanning trees of the VN and computes a shortest-path based routing of the VLinks such that at least one of the spanning trees survives after an SLink failure. In contrast, our solution is generic, *i.e.*, does not assume any specific property of the VN and SN, and can ensure connectivity in presence of multiple SLink failures.

Several research works from IP-over-WDM literature ensure survivability through IP link augmentation [12], [13], [19], [20]. However, all of these works focus on the single failure resiliency and cannot be generalized to multiple failures. In contrast, Thulasiraman *et al.* propose an augmentation strategy for ensuring survivability under k SLink failures [8]. They propose to augment VLinks until a complete subgraph of $k+1$ VNodes is constructed and the remaining VNodes are $k+1$ edge connected to the subgraph. Their solution maps any k of the VLinks incident to a VNode onto disjoint paths. This approach requires higher number of VLinks to be augmented and more disjointness constraints on the SN than our approach.

III. Preliminaries

The subsequent sections build upon the background, definitions, and assumptions presented in this section.

A. System Model

1) Substrate Network: We represent the Substrate Network (SN) as an undirected graph, $G = (V, E)$, where V and E denote the set of Substrate Nodes (SNodes) and Substrate Links (SLinks), respectively. The set of neighbors of an SNode $u \in V$ is denoted by $\mathcal{N}(u)$. Bandwidth capacity of an SLink $(u, v) \in E$ is b_{uv}, while the cost of allocating one unit of bandwidth in (u, v) is C_{uv}.

2) Virtual Network: A VN is represented as an undirected graph $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V}$ and $\bar{E}$ denote the set of Virtual Nodes (VNodes) and Virtual Links (VLinks), respectively. The neighbors of a VNode $\bar{v} \in \bar{V}$ is denoted by $\mathcal{N}(\bar{v})$. Each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ has bandwidth requirement $b_{\bar{u}\bar{v}}$. Each VNode $\bar{u} \in \bar{V}$ has a location constraint, $L(\bar{u}) \subseteq V$, that denotes the set of SNodes where $\bar{u}$ can be embedded.

B. Design Choices

The first condition for surviving k SLink failures is that a VN must be $k+1$ edge connected. However, if the input VN $\bar{G}$ does not have such connectivity, we will need to augment

TABLE I
NOTATION TABLE

$G = (V, E)$	Substrate Network (SN)
$\bar{G} = (\bar{V}, \bar{E})$	Virtual Network (VN)
$\hat{G} = (\hat{V}, \hat{E})$	k-protected VN
$\hat{G}_k = (\hat{V}_k, \hat{E}_k)$	k-protected component of a VN $\bar{G}$
$\hat{G}_k \odot \hat{v}$	An expansion of $\hat{G}_k$ towards $\hat{v}$
$\chi^{\hat{u}\hat{v}}$	Conflicting set of a VLink $(\hat{u}, \hat{v})$
$\chi^{\hat{u}\hat{v}}_{\odot}$	Conflicting set of a VLink $(\hat{u}, \hat{v})$ during expansion
$\chi^{\hat{G}}$	Conflicting set of a VN $\hat{G}$
Q^{uv}	A path between SNodes u and v in G
$P^{\hat{u}\hat{v}}$	A path between VNodes $\hat{u}$ and $\hat{v}$ in $\hat{G}$
$\mathscr{P}^{\hat{u}\hat{v}}$	Set of edge-disjoint paths between $\hat{u}$ and $\hat{v}$ in $\hat{G}$
$\mathrm{p}_i^{\hat{u}\hat{v}}$	ith edge-disjoint shortest path from $\hat{u}$ to $\hat{v}$ in $\hat{G}$
$\mathcal{P}^{\hat{G}_k\hat{v}}$	Set of edge-disjoint shortest paths from $\hat{v}$ to $\hat{G}_k$

$\bar{G}$ with additional VLinks. This augmentation can be done in two ways: i) VLinks can be augmented between arbitrary pair of VNodes to ensure $k+1$ edge connectivity, which is a well studied problem [19], [20]; ii) the other way is to augment only parallel VLinks between adjacent VNodes in $\bar{G}$ [12], [13]. Arbitrary augmentation can ensure $k+1$ edge connectivity by introducing minimal number of VLinks, but this approach will change the input VN topology. Although parallel VLink augmentation may not be minimal in terms of resource usage, it does not change the input VN topology. From VN user perspective, it is very important to preserve the input VN topology. Hence, we opt for the second alternative, *i.e.*, to augment parallel VLinks only.

We use the term k-protected VN, $\hat{G} = (\hat{V}, \hat{E})$, to represent a VN that is made $k+1$ edge connected by adding parallel VLinks to an input VN, $\bar{G} = (\bar{V}, \bar{E})$. Here, $\hat{V} = \bar{V}$ and $\hat{E} = \bar{E} \cup \tilde{E}$ where $\tilde{E}$ is the set of parallel VLinks to be added. Determining the capacity of the parallel VLinks (in $\tilde{E}$) as well as the amount of spare capacity to be reserved for the input VLinks (in $\bar{E}$) in order to guarantee full bandwidth of a failed VLink is a separate problem of its own [21]. In this work, we assume that the capacity of a parallel VLink will be the same as the capacity of the input VLink it augments.

C. Definitions

Definition 1. ***k-protected component:*** *A k-protected component of a graph $\bar{G}$ is a multi-graph $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$, where $\hat{V}_k \subseteq \bar{V}$, $\hat{E}_k = \bar{E}_k \cup \tilde{E}_k$, $\bar{E}_k \subseteq \bar{E}$, $\tilde{E}_k \subseteq \tilde{E}$ and $\tilde{E}_k$ is a set of parallel VLinks augmented in such a way that simultaneous removal of k arbitrary VLinks in $\hat{G}_k$ will not partition $\hat{G}_k$.*

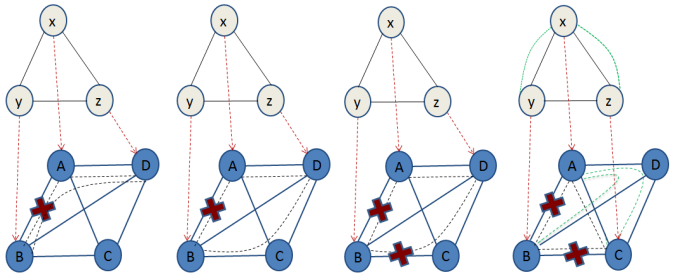
Definition 2. ***Conflicting VLinks:*** *Two VLinks are considered as conflicting if they must be embedded on edge-disjoint substrate paths in order to ensure $k+1$ edge connectivity.*

Definition 3. ***Conflicting set:*** *A conflicting set of a VLink $(\hat{u}, \hat{v})$, denoted by $\chi^{\hat{u}\hat{v}}$, is the set of VLinks in $\hat{E}$ those are conflicting with $(\hat{u}, \hat{v})$. A Conflicting set of a VN $\hat{G} = (\hat{V}, \hat{E})$, denoted by $\chi^{\hat{G}}$, is defined as* $\chi^{\hat{G}} = \bigcup_{\forall(\hat{u},\hat{v})\in\hat{E}} \chi^{\hat{u}\hat{v}}$.

D. CoViNE Problem Statement

Given an SN $G = (V, E)$, a VN $\bar{G} = (\bar{V}, \bar{E})$, and location constraints $L(\bar{u})$ for all $\bar{u} \in \bar{V}$ find an embedding that

- provides a function $f : \bar{V} \to V$ to map every VNode $\bar{u} \in \bar{V}$ to exactly one SNode $u \in V$ while satisfying the location constraint and without any overlap, i.e., $\forall \bar{u}, \bar{v} \in \bar{V} \wedge \bar{u} \neq \bar{v} \implies f(\bar{u}) \neq f(\bar{v})$ and $\forall \bar{u} \in \bar{V}\ \ f(\bar{u}) \in L(\bar{u})$,
- provides a function $g : \bar{E} \to 2^E$ to map each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ to a substrate path $Q^{f(\bar{u})f(\bar{v})}$ with sufficient bandwidth to satisfy the VLink demand $b_{\bar{u}\bar{v}}$,
- ensures the connectivity in $\bar{G}$ in presence of up to k SLink failures in G,
- minimizes the total cost of embedding in terms of substrate bandwidth consumption.



(a) Single Failure (b) Double Failure

Fig. 1. CoViNE examples

We illustrate CoViNE examples for different failure scenarios in Fig. 1. In these examples, xyz is the VN and $ABCD$ is the SN. The arrow from a VNode to an SNode defines node mapping and the dotted lines between SNodes define link mapping. To survive single $(k = 1)$ failure, the VN must be 2 edge-connected. Since xyz VN is already 2 edge-connected, no augmentation is required. Fig. 1(a) shows an un-survivable embedding (on the left) and a survivable embedding (on the right) of the xyz VN. They differ in satisfying disjointness constraints. The embedding on the left satisfies no disjointness constraint, hence VLinks (x, y) and (y, z) share an SLink (A, B). Upon the failure of (A, B), both VLinks fail, and VNode y is disconnected from the rest of the VN. The embedding on the right adheres to the disjointness constraints, hence no sharing of SLinks is possible. Even though SLink (A, B) and correspondingly VLink (x, y) fail, the VN remains connected.

Fig. 1(b) exhibits the double $(k = 2)$ failure scenario for the same VN and SN topology. To survive double failures, the VN must have 3 edge-connectivity which is absent in the xyz VN. The embedding on the left demonstrates that even an edge-disjoint embedding of the VN results in an un-survivable embedding for double link failures due to the lack of necessary edge-connectivity in the VN. For the embedding on the right, the VN has necessary edge-connectivity through augmentation of green colored VLinks, and embedding is done adhering to the disjointness constraints resulting in a survivable embedding. It is to be noted that for the augmented VN on the right, not all the VLinks need to be disjoint with each other, hence there are some sharing of SLinks.

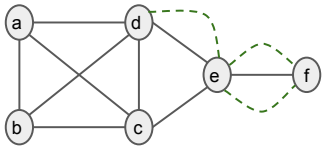


Fig. 2. The VN with only solid edges is the input VN, $\bar{G}$. The VN with both solid edges ($\bar{E}$) and dashed edges ($\hat{E}$) is the 2-protected VN, $\hat{G}$. Any subgraph of $\hat{G}$ having 3 edge connectivity is $\hat{G}_2$.

IV. Problem Formulation

In this section, we propose a simple but efficient mechanism for computing *Conflicting set* of a VN (§ IV-A). We also show that this mechanism helps to transform a given VN $\bar{G}$ to a k-protected VN $\hat{G}$ by augmenting parallel VLinks only (§ IV-B).

A. Conflicting Set Computation

In this subsection, we assume that the VNs are k-protected, while the mechanism for transforming an arbitrary VN to a k-protected VN is discussed in the next subsection. In order to remain connected in presence of k SLink failures, the embedding algorithm must ensure $k+1$ edge connectivity between SNodes $f(\bar{u})$ and $f(\bar{v})$ for every pair of VNodes $\hat{u} \in \hat{V}$ and $\hat{v} \in \hat{V}$ of the k-protected VN $\hat{G}$. This can be achieved if the VLinks of every edge-cut in $\hat{G}$ are embedded on at least $k+1$ edge-disjoint paths in G. Since there are exponential number of edge-cuts in $\hat{G}$ and there are combinatorial number of ways of choosing $k+1$ conflicting VLinks from an edge-cut in $\hat{G}$, the number of possibilities for computing a conflicting set of $\hat{G}$ is enormous. However, an *optimal conflicting set* of $\hat{G}$ is one that ensures $k+1$ edge connectivity of the embedding of $\hat{G}$ while minimizing disjoint path requirement in the embedding. This can be achieved by finding the minimum number of partitions of the VLinks of $\hat{E}$ such that the VLinks in a partition are not conflicting with each other. Since the VLinks in a partition do not impose any disjointness constraint, minimizing the number of partitions will yield optimal conflicting set. Computing the optimal conflicting set of a VN is NP-complete since it can be reduced to the the well-known *Minimum vertex coloring* problem.[1] The following results provide a basis of our heuristic algorithm for computing the conflicting set of a VN $\hat{G}$.

Theorem 1. *([22]) The size of the minimum edge-cut for two distinct VNodes $\hat{u}, \hat{v} \in \hat{G}$ is equal to the maximum number of edge-disjoint paths between $\hat{u}$ and $\hat{v}$ in $\hat{G}$.*

According to Theorem 1, also known as Menger's Theorem, any pair of VNodes $\hat{u}$ and $\hat{v}$ in $\hat{G}$ will remain connected in presence of k SLink failures, if at least one of the edge-disjoint paths $P_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}}$ remains intact. This can be achieved by mapping any $k+1$ paths in $\mathscr{P}^{\hat{u}\hat{v}}$ into $k+1$ edge-disjoint paths in the SN. There are a combinatorial number of ways of choosing these $k+1$ edge-disjoint paths between $\hat{u}$ and $\hat{v}$. If $\mathscr{P}_1^{\hat{u}\hat{v}} = P_1^{\hat{u}\hat{v}}, P_2^{\hat{u}\hat{v}}, ..., P_{k+1}^{\hat{u}\hat{v}}$ is one possible combination chosen to have edge-disjoint mapping, two VLinks $(\hat{x}, \hat{y}) \in P_i^{\hat{u}\hat{v}}$ and $(\hat{w}, \hat{z}) \in P_j^{\hat{u}\hat{v}}$, s.t. $x \neq w$ and $y \neq z$, cannot share an SLink in their mappings. Therefore, a VLink $(\hat{x}, \hat{y}) \in P_i^{\hat{u}\hat{v}}$ is conflicting with all other VLinks present in the paths in $\mathscr{P}_1^{\hat{u}\hat{v}} \setminus P_i^{\hat{u}\hat{v}}$, leading to $|\chi^{\hat{x}\hat{y}}| = \sum_{P_i^{\hat{u}\hat{v}} \in \mathscr{P}_1^{\hat{u}\hat{v}} \wedge (\hat{x},\hat{y}) \notin P_i^{\hat{u}\hat{v}}} |P_i^{\hat{u}\hat{v}}|$. For example, in Fig. 2, VNodes a and b will remain connected in presence of 2 SLink failures if the VLinks on paths $P_1^{ab} = (a,b)$, $P_2^{ab} = \{(a,d),(d,c),(c,b)\}$, and $P_3^{ab} = \{(a,c),(c,e),(e,d),(d,b)\}$ are mapped to disjoint SN paths. Hence, $\chi^{ab} = P_2^{ab} \cup P_3^{ab}$.

We now discuss some heuristics to reduce the above computation. First, we can ensure connectivity in $\hat{G}$ by ensuring connectivity in a minimum spanning tree $\hat{T}$ of $\hat{G}$. In this case, we need to compute $k+1$ edge-disjoint paths only for the $|\hat{V}|-1$ VLinks in $\hat{T}$, as opposed to considering all the VLinks in $\hat{G}$. For the VN in Fig. 2, $k+1$ edge-disjoint path computations are required for the VLinks in $\hat{T} = \{(a,b),(a,c),(c,d),(d,e),(e,f)\}$ instead of all the 12 VLinks in $\hat{G}$. Second, instead of arbitrarily picking $k+1$ edge-disjoint paths from $\mathscr{P}^{\hat{u}\hat{v}}$, we can pick the first $k+1$ edge-disjoint shortest paths between $\hat{u}$ and $\hat{v}$. Thus, the size of the conflicting set of a VLink $(\hat{u},\hat{v})$ in $\hat{T}$ becomes $|\chi^{\hat{u}\hat{v}}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}} \wedge (\hat{u},\hat{v}) \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$, where $\mathtt{p}_i^{\hat{u}\hat{v}}$ is the ith edge-disjoint shortest path between two adjacent VNodes $\hat{u}$ and $\hat{v}$. This method yields smaller conflicting set $\chi^{ab} = \mathtt{p}_2^{ab} \cup \mathtt{p}_3^{ab}$, where $\mathtt{p}_2^{ab} = \{(a,c),(c,b)\}$, and $\mathtt{p}_3^{ab} = \{(a,d),(d,b)\}$ in Fig. 2.

Definition 4. ***Expansion Operator*** $\odot$*: Given a k-protected component $\hat{G}_k$ of a VN $\hat{G}$ and a VNode $\hat{v}$ s.t., $\hat{v} \in \hat{V} \setminus \hat{V}_k$ and $\exists \hat{u} \in \hat{V}_k, \hat{v} \in \mathcal{N}(\hat{u})$, we define $\hat{G}_k \odot \hat{v}$ as an expansion of $\hat{G}_k$ generated by adding $\hat{v}$ and all the incident VLinks on $\hat{v}$ from any VNode in $\hat{G}_k$. Mathematically,*

$$\hat{G}_k \odot \hat{v} = (\hat{V}_k \cup \{\hat{v}\}, \hat{E}_k \cup \{(\hat{u},\hat{v}) | \hat{u} \in \hat{V}_k, \hat{u} \in \mathcal{N}(\hat{v})\})$$

Definition 5. ***EDSP*** $\mathcal{P}^{\hat{G}_k\hat{v}}$***:*** *We define EDSP as a set of Edge-Disjoint Shortest Paths $\mathcal{P}^{\hat{G}_k\hat{v}} = \{\mathtt{p}_i^{\hat{x}\hat{v}}\}$ between $\hat{G}_k$ and a VNode $\hat{v} \in \hat{V} \setminus \hat{V}_k$ s.t. $\hat{x} \in \hat{V}_k$ and all $\mathtt{p}_i^{\hat{x}\hat{v}}$ terminate as the first VNode $\hat{x}$ in $\hat{V}_k$ is encountered, i.e., the only VNode from $\hat{V}_k$ that is on $\mathtt{p}_i^{\hat{x}\hat{v}}$ is $\hat{x}$.*

Observation 1: Using the expansion lemma, it can be shown that $\hat{G}_k \odot \hat{v}$ is a k-protected component if and only if there exists $k+1$ edge-disjoint paths from $\hat{G}_k$ to $\hat{v}$ in $\hat{G}$ [8]. Furthermore, it can be intuitively observed that in $\hat{G}_k \odot \hat{v}$, any $k+1$ EDSPs in $\mathcal{P}^{\hat{G}_k\hat{v}}$ will not contain a VLink from $\hat{E}_k$.

Lemma 1. *In an expansion $\hat{G}_k \odot \hat{v}$, the size of the conflicting set of a VLink $(\hat{u},\hat{v}) \in \hat{E} \setminus \hat{E}_k$ is $|\chi_{\odot}^{\hat{u}\hat{v}}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v}) \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$, where $\hat{u} \in \hat{V}_k$ and $\hat{v} \in \mathcal{N}(\hat{u})$.*

Proof. For the embedding of $\hat{G}_k \odot \hat{v}$ on G to remain connected in presence of k SLink failures, we need to satisfy two conditions: i) at least $k+1$ edge-disjoint paths from $\hat{v}$ to $\hat{G}_k$ exist (*i.e.*, $|\mathcal{P}^{\hat{G}_k\hat{v}}| \geq k+1$), and ii) all of these paths are embedded on $k+1$ edge-disjoint paths in G. Therefore, the VLink $(\hat{u},\hat{v})$ is conflicting with all the VLinks in the first $k+1$ edge-disjoint shortest paths in $\mathcal{P}^{\hat{G}_k\hat{v}} \setminus \mathtt{p}_i^{\hat{u}\hat{v}}$, where $(\hat{u},\hat{v}) \in \mathtt{p}_i^{\hat{u}\hat{v}}$. This leads to $|\chi_{\odot}^{\hat{u}\hat{v}}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v}) \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$. □

[1] *Minimum vertex coloring* is to color the vertices of a graph with a minimum number of colors so that adjacent vertices are of different colors.

Theorem 2. *In comparison to computing conflicting set $\chi^{\hat{u}\hat{v}}$ independently for a VLink $(\hat{u},\hat{v}) \in \hat{E}$, computing conflicting set for $(\hat{u},\hat{v})$ through the expansion $\hat{G}_k \odot \hat{v}$ will generate conflicting sets of lesser or equal size.*

Proof. Let's consider two VNodes $\hat{u} \in \hat{V}_k$ and $\hat{v} \in \hat{V} \setminus \hat{V}_k$ s.t. $\hat{v} \in \mathcal{N}(\hat{u})$. When computed independently, the size of the conflicting set of $(\hat{u},\hat{v})$ is $|\chi^{\hat{u}\hat{v}}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}} \wedge (\hat{u},\hat{v}) \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$. On other hand, when we construct conflicting set through the expansion, $\hat{G}_k \odot \hat{v}$, the size of the conflicting set of the VLink $(\hat{u},\hat{v}) \in \hat{E} \setminus \hat{E}_k$ is $|\chi_{\odot}^{\hat{u}\hat{v}}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k \hat{v}} \wedge (\hat{u},\hat{v}) \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$ (as proven in Lemma 1). In the beginning, when $\hat{G}_k$ contains only one VNode i.e. $|\hat{V}_k| = 1$, it is obvious that $|\chi^{\hat{u}\hat{v}}| = |\chi_{\odot}^{\hat{u}\hat{v}}|$. For $|\hat{V}_k| > 1$, consider $\hat{x} \in \hat{V}_k$ s.t. $\exists \mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}}$ contains $\hat{x}$ and $\mathtt{p}_j^{\hat{x}\hat{v}} \in \mathcal{P}^{\hat{G}_k \hat{v}}$. Since $\mathtt{p}_i^{\hat{u}\hat{v}}$ contains $\hat{x}$, according to the optimal substructure property of shortest path, we get $\mathtt{p}_i^{\hat{u}\hat{v}} = \mathtt{p}_i^{\hat{u}\hat{x}} || \mathtt{p}_j^{\hat{x}\hat{v}}$, assuming $||$ is the path concatenation operator. Thus, $|\mathtt{p}_j^{\hat{x}\hat{v}}| < |\mathtt{p}_i^{\hat{u}\hat{v}}|$ resulting into $|\chi_{\odot}^{\hat{u}\hat{v}}| < |\chi^{\hat{u}\hat{v}}|$. If such an $\hat{x}$ is not found, we can assume $\hat{x} = \hat{u}$ and in that case $\mathtt{p}_j^{\hat{x}\hat{v}} = \mathtt{p}_i^{\hat{u}\hat{v}}$ yielding $|\chi_{\odot}^{\hat{u}\hat{v}}| = |\chi^{\hat{u}\hat{v}}|$. Hence, $|\chi_{\odot}^{\hat{u}\hat{v}}| \leq |\chi^{\hat{u}\hat{v}}|$. □

As an example of Theorem 2, let us consider the VLink (d,e) in Fig. 2 and the VN needs to survive against single SLink failure. If we compute independently, we get $\chi^{de} = \{(d,c),(c,e)\}$. When we compute through expansion $\hat{G}_1 \odot e$ where $\hat{V}_1 = \{a,b,c,d\}$, we get $\chi^{de} = \{(c,e)\}$.

B. VLink Augmentation

As described in § III-B, we may need to augment a given VN $\bar{G}$ with parallel VLinks in order to make it a k-protected VN $\hat{G} = (\hat{V},\hat{E})$. Now, the challenge here is to minimize the number of augmented parallel VLinks. We use Menger's Theorem [22] to find the pair of VNodes with less than $k+1$ edge connectivity and add parallel VLinks as needed. Assume that for each pair of adjacent VNodes $\bar{u},\bar{v} \in \bar{V}$ there are at least m edge-disjoint paths in $\bar{G}$. If $m \geq k+1$, $\bar{G}$ is at least $k+1$ edge-connected, hence no augmentation is needed. If $m < k+1$, we need to add $k+1-m$ parallel VLinks between $\bar{u}$ and $\bar{v}$. In general, $max(0, k+1-m)$ parallel VLinks are needed for each pair of adjacent VNodes. For instance, a VN should be 3 edge connected to survive 2 SLink failures. Since there are 2 edge-disjoint paths between d and e in Fig. 2, we add a parallel VLink. Similarly, we add two parallel VLinks between e and f to make the VN 3 edge connected. No augmentation is required for the rest of the adjacent pair of VNodes. It can be easily shown that the number of parallel VLinks to be augmented remains the same during the expansion, $\hat{G}_k \odot \bar{v}$. In other words, if there are $\hat{m}$ edge-disjoint paths from $\hat{G}_k$ to $\bar{v}$ in $\bar{G}$, augmentation of $max(0, k+1-\hat{m})$ parallel VLinks is needed to ensure the $k+1$ edge connectivity between $\hat{u}$ and $\bar{v}$, where $\hat{u} \in \hat{V}_k$ and $\bar{v} \in \mathcal{N}(\hat{u})$.

V. Heuristic Algorithm for Conflicting Set

Given the NP-complete nature of computing optimal conflicting set (§ IV), we propose a heuristic algorithm (Algorithm 1) for computing conflicting sets within a reasonable time. Algorithm 1 starts with a k-protected component, $\hat{G}_k$, containing an arbitrary VNode $\bar{u} \in \bar{V}$. The algorithm then includes all of $\bar{u}$'s neighbors $\bar{v} \in \mathcal{N}(\bar{u})$ to $\hat{V}_k$. This process is repeated until all the VNodes of $\bar{G}$ are added to $\hat{G}_k$. For each $\bar{v}$, the algorithm computes $k+1$ EDSPs, $\mathcal{P}^{\hat{G}_k \bar{v}}$ between $\hat{G}_k$ and $\bar{v}$ using the procedure *COMPUTE-EDSP* (Line 8). This procedure initially includes the VLink, $(\bar{u},\bar{v})$ as the first shortest path $\mathtt{p}_1^{\hat{G}_k \bar{v}}$ to $\mathcal{P}^{\hat{G}_k \bar{v}}$. It then invokes *Dijkstra's shortest path* algorithm k times to compute $\mathtt{p}_i^{\hat{G}_k \bar{v}}$, the ith EDSP between $\hat{G}_k$ and $\bar{v}$. After computing each $\mathtt{p}_i^{\hat{G}_k \bar{v}}$, all the VLinks present in $\mathtt{p}_i^{\hat{G}_k \bar{v}}$ are removed from $\bar{G}$ in order to ensure the edge-disjointness of the later paths. Although we use *Dijkstra's shortest path* algorithm repeatedly to compute $k+1$ EDSPs, other algorithms from the literature can be used for this purpose [9]. If the number of computed EDSPs is less than $k+1$, Algorithm 1 adds $k+1-|\mathcal{P}^{\hat{G}_k \bar{v}}|$ parallel VLinks between $\bar{u}$ and $\bar{v}$ (Line 10). The ith parallel VLink is denoted by $(\bar{u},\bar{v})^i$, and constitutes the $(|\mathcal{P}^{\hat{G}_k \bar{v}}|+i)$th EDSP between $\hat{G}_k$ and $\bar{v}$. Finally, Algorithm 1 updates the conflicting sets of the corresponding VLinks as described in Lemma 1 (Line 13).

Algorithm 1 Compute Conflicting Sets

```
1: function COMPUTE-CONFLICTING-SETS(Ḡ)
2:     ∀(ū, v̄) ∈ Ē: χ^{ūv̄} ← φ, Q ← φ
3:     ∃v̄ ∈ V̄ : Ĝ_k ← ({v̄}, φ) // v̄ is an arbitrary VNode
4:     ENQUEUE(Q, v̄)
5:     while Q is not empty do
6:         ū ← DEQUEUE(Q)
7:         for all v̄ ∈ N(ū) and v̄ ∉ Ĝ_k do
8:             P^{Ĝ_k v̄} ← COMPUTE-EDSP(Ḡ, Ĝ_k, ū, v̄, k+1)
9:             for i = 1 → (k+1−|P^{Ĝ_k v̄}|) do
10:                Ē ← Ē ∪ (ū, v̄)^i, P^{Ĝ_k v̄} ← P^{Ĝ_k v̄} ∪ (ū, v̄)^i
11:            end for
12:            ∀(x̄, ȳ) ∈ p_i^{Ĝ_k v̄}, p_i^{Ĝ_k v̄} ∈ P^{Ĝ_k v̄} :
13:              χ^{x̄ȳ} ← χ^{x̄ȳ} ∪ {(s̄, t̄) ∈ p_j^{Ĝ_k v̄} | p_j^{Ĝ_k v̄} ∈ P^{Ĝ_k v̄} ∧ i ≠ j}
14:            Ĝ_k ← Ĝ_k ⊙ v̄
15:            ENQUEUE(Q, v̄)
16:        end for
17:    end while
18: return χ^Ḡ
19: end function
```

The time complexity of the heuristic algorithm is dominated by the *COMPUTE-EDSP* procedure, which invokes *Dijkstra's shortest path* algorithm k times. The time complexity of *Dijkstra's shortest path* algorithm based on a min-priority queue is $O(|\bar{E}| + |\bar{V}| \log |\bar{V}|)$. Since *COMPUTE-EDSP* is invoked $O(|\bar{V}||\mathcal{N}(\bar{u})|)$ times, the running time of Algorithm 1 becomes $O(k|\bar{V}||\mathcal{N}(\bar{u})|(|\bar{E}| + |\bar{V}| \log |\bar{V}|))$.

VI. ILP Formulation for CoViNE

In this section, we present an Integer Linear Programming (ILP) formulation for CoViNE. The ILP minimizes the total cost of provisioning bandwidth for the VLinks of a VN, $\hat{G}$.

We represent the location constraint $L(\hat{u}) \subseteq V$ of $\hat{u} \in \hat{V}$ with the binary variable $\ell_{\hat{u}u}$ defined as follows:

$$\ell_{\hat{u}u} = \begin{cases} 1 & \text{if } \hat{u} \in \hat{V} \text{ can be mapped to } u \in V, \\ 0 & \text{otherwise.} \end{cases}$$

A. Decision Variables

A VLink is mapped to a path in SN. The following decision variable indicates the mapping between a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ and an SLink $(u, v) \in E$.

$$x_{uv}^{\hat{u}\hat{v}} = \begin{cases} 1 & \text{if } (\hat{u}, \hat{v}) \in \hat{E} \text{ is mapped to } (u, v) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The following variable represents VNode mapping:

$$y_{\hat{u}u} = \begin{cases} 1 & \text{if } \hat{u} \in \hat{V} \text{ is mapped to } u \in V, \\ 0 & \text{otherwise.} \end{cases}$$

B. Constraints

1) VLink Mapping Constraints: Our VLink mapping constraints are as follows:

$$\forall (\hat{u}, \hat{v}) \in \hat{E} : \sum_{\forall (u,v) \in E} x_{uv}^{\hat{u}\hat{v}} \geq 1 \tag{1}$$

$$\forall (u, v) \in E : \sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} x_{uv}^{\hat{u}\hat{v}} \times b_{\hat{u}\hat{v}} \leq b_{uv} \tag{2}$$

$$\forall \hat{u}, \hat{v} \in \hat{V}, \forall u \in V : \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{\hat{u}\hat{v}} - x_{vu}^{\hat{u}\hat{v}}) = y_{\hat{u}u} - y_{\hat{v}u} \tag{3}$$

(1) ensures that each VLink is mapped to a non-empty set of SLinks and no VLink is left unmapped. (2) ensures that an SLink is not assigned VLink demands that exceeds the SLink's capacity. Finally, (3) ensures that the in-flow and out-flow of each SNode is equal except at the SNodes where the endpoints of a VLink are mapped following the constraint in [23].

2) VNode Mapping Constraints: We map the VNodes according to the location constraint as described in (4). We also ensure that a VNode is mapped to exactly one SNode by (5). Finally, (6) ensures that an SNode does not host more than one VNode from the same VN. The VNode mapping follows from the VLink mapping since we do not have any VNode mapping cost.

$$\forall \hat{u} \in \hat{V}, \forall u \in V : y_{\hat{u}u} \leq \ell_{\hat{u}u} \tag{4}$$

$$\forall \hat{u} \in \hat{V} : \sum_{u \in V} y_{\hat{u}u} = 1 \tag{5}$$

$$\forall u \in V : \sum_{\hat{u} \in \hat{V}} y_{\hat{u}u} \leq 1 \tag{6}$$

3) Disjointness Constraints: To ensure the desired survivability of CoViNE, a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ should never share an SLink with it's conflicting VLinks in $\chi^{\hat{u}\hat{v}}$ in their mappings. This disjointness requirement is ensured with the following constraint $\forall (u, v) \in E, \forall (\hat{u}, \hat{v}) \in \hat{E}, \forall (\hat{a}, \hat{b}) \in \chi^{\hat{u}\hat{v}}$:

$$x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{a}\hat{b}} + x_{vu}^{\hat{a}\hat{b}} \leq 1 \tag{7}$$

C. Objective Function

Our objective is to minimize the bandwidth provisioning cost over all the SLinks used by the mappings of all the VLinks of a VN, $\hat{G}$. Given that C_{uv} is the cost of allocating unit bandwidth on SLink $(u, v) \in E$, we have the following objective function for our ILP:

$$\text{minimize} \left(\sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} \sum_{\forall (u,v) \in E} x_{uv}^{\hat{u}\hat{v}} \times C_{uv} \times b_{\hat{u}\hat{v}} \right)$$

VII. Heuristic Algorithm for CoViNE

The ILP formulation presented in § VI cannot solve larger instances of the problem due to the limitation of LP solvers. Hence, we propose a heuristic algorithm (Algorithm 2) to produce near-optimal solutions within reasonable time limit. Algorithm 2 embeds $\hat{G}$ while ensuring the disjointness constraint imposed by $\chi^{\hat{G}}$ and minimizing the total cost of embedding according to the objective function in § VI-C.

Algorithm 2 computes two functions, $nmap$ and $emap$, which represent the VNode and VLink mapping of $\hat{G}$ on G, respectively. Since there is no cost associated with VNode mapping, a VLink mapping that minimizes total cost determines the VNode mapping. Algorithm 2 first sorts the VNodes $\hat{u} \in \hat{V}$ in decreasing order of the sum of conflicting set sizes of incident VLinks. This sorted list of VNodes is represented by $\hat{\mathcal{V}}$. A VNode with VLinks having larger conflicting sets becomes too constrained to be mapped to a suitable SNode, hence, Algorithm 2 tries to map VNodes in the order of $\hat{\mathcal{V}}$.

For each VNode $\hat{u} \in \hat{\mathcal{V}}$, Algorithm 2 searches for an unallocated SNode in $\hat{u}$'s location constraint set, $L(\hat{u})$, which yields a feasible mapping while minimizing the cost. To embed $\hat{u}$, Algorithm 2 loops through each candidate SNode $l \in L(\hat{u})$ (Line $5-14$), to first temporarily map $\hat{u}$ to l (Line 6). Then

Algorithm 2 VN-Embedding

```
1: function VN-EMBEDDING(G, Ĝ)
2:     V̂ ← Sort û ∈ V̂ in decreasing order of Σ_{∀v̂∈N(û)} |χ^{ûv̂}|
3:     for all û ∈ V̂ do
4:         Candidate ← φ
5:         for all l ∈ L(û) do
6:             Add mapping û → l to nmap
7:             E ← Sort (û, v̂) ∈ Ê in decreasing order of |χ^{ûv̂}|
8:             ∀(û, v̂) ∈ E such that emap(û, v̂) = φ
9:                 P[(û, v̂)] ← VLINK-MAP(G, Ĝ, (û, v̂))
10:            if Σ_{∀(u,v)∈E} cost(P[(û, v̂)]) is minimum then
11:                M ← P, Candidate ← l
12:            end if
13:            nmap(û) ← φ, ∀(û, v̂) ∈ E: emap(û, v̂) ← φ
14:        end for
15:        if Candidate ≠ φ then
16:            Add mapping û → Candidate to nmap
17:            ∀(û, v̂) ∈ E and nmap(û) ≠ φ and nmap(v̂) ≠ φ:
18:                Add mapping (û, v̂) → M[(û, v̂)] to emap
19:        else
20:            return No Solution Found
21:        end if
22:    end for
23: return {nmap, emap}
24: end function
```

Algorithm 3 VLink-Map

```
1: function VLINK-MAP(G, Ĝ, (û, v̂))
2:     p^{ûv̂} ← φ
3:     ∀(ŝ, t̂) ∈ χ^{ûv̂} :
4:         E ← E − {(a, b) ∈ E|(ŝ, t̂) is mapped to (a, b)}
5:     if nmap(û) ≠ φ ∧ nmap(v̂) ≠ φ then
6:         Q^{nmap(û)nmap(v̂)} ← MCP(G, nmap(û), nmap(v̂), b_{ûv̂})
7:     else if nmap(x̂) = φ ∧ nmap(ŷ) ≠ φ s.t. x̂, ŷ ∈ {û, v̂},
   ŷ ≠ x̂ then
8:         Q^{nmap(û)nmap(v̂)} ←
9:             min_{∀l∈L(x̂)} {MCP(G, nmap(ŷ), l, b_{ŷl})}
10:    end if
11:    if Q^{nmap(û)nmap(v̂)} ≠ φ then
12:        Add mapping (û, v̂) → Q^{nmap(û)nmap(v̂)} to emap
13:    end if
14: return Q^{nmap(û)nmap(v̂)}
15: end function
```

the algorithm tries to embed all the VLinks incident to $\hat{u}$. The *VLINK-MAP* (Algorithm 3) procedure is invoked to find the mapping for each such VLink (Line 9). VLinks incident to $\hat{u}$ are processed in the decreasing order of their conflicting set sizes to maximize the chances of finding substrate paths that can satisfy the disjointness constraint enforced by the conflicting sets. Algorithm 2 finally embeds $\hat{u}$ to the l that leads to a feasible mapping for all the VLinks incident to $\hat{u}$ and yields the minimum embedding cost. The algorithm fails, if no such feasible l is found. Once a VNode $\bar{u}$ has been finally mapped, Algorithm 2 creates the final mapping for only those VLinks incident to $\hat{u}$ whose both endpoints are already finally mapped (Line $17-18$). The mappings of other VLinks incident on $\hat{u}$ are finalized when their unmapped endpoints are mapped. We now describe the *VLINK-MAP* (Algorithm 3) procedure for finding the mapping of a VLink, $(\hat{u}, \hat{v})$. First we remove all the SLinks used by the mapping of all the VLinks in $\chi^{\hat{u}\hat{v}}$ to satisfy the disjointness constraint (Line $3-4$). Then, we compute mapping for $(\hat{u}, \hat{v})$ by considering the following two cases: (i) both endpoints of $(\hat{u}, \hat{v})$ have already been mapped to some SNodes (Line 5). In this case, we find a minimum cost path between $nmap(\hat{u})$ and $nmap(\hat{v})$ with capacity at least $b_{\hat{u}\hat{v}}$ in G; (ii) only $\hat{u}$ (or $\hat{v}$) is mapped and the other endpoint $\hat{v}$ (or $\hat{u}$) has not been mapped (Line 7). In this case, we compute the minimum cost path between $nmap(\hat{u})$ (or $nmap(\hat{v})$) and all possible locations for the unmapped VNode $\hat{v}$ (or $\hat{u}$), $l \in L(\hat{v})$ (or $L(\hat{u})$) with at least $b_{\hat{u}\hat{v}}$ capacity. $(\hat{u}, \hat{v})$ is temporarily mapped to this path and the mapping is added to $emap$ (Line 12). We modified *Dijkstra's shortest path* algorithm to consider link capacities while computing the minimum cost path (MCP procedure call in Algorithm 3). The cost for each SLink is set $(u, v) \in E$ to $C_{uv} \times b_{\hat{u}\hat{v}}$, where $b_{\hat{u}\hat{v}}$ is the bandwidth requirement of the VLink to be embedded.

The most expensive step of Algorithm 2 is the *VLINK-MAP* function, which invokes *Dijkstra's shortest path* algorithm on the SN requiring $O(|E|+|V|\log|V|)$ time. Since *VLINK-MAP* is invoked $O(|\hat{V}||L(\hat{u})||\mathcal{N}(\hat{u})|)$ times, the running time of Algorithm 2 becomes $O(|\hat{V}||L(\hat{u})||\mathcal{N}(\hat{u})|(|E|+|V|\log|V|))$.

VIII. Evaluation

A. Compared Approaches

We compare six approaches (Table II) that combine different strategies for computing disjointness constraint and VN embedding. We have chosen single failure ($k = 1$) and double failure ($k = 2$) scenarios, since the possibility of more than two simultaneous link failures is very low [7], [3]. The first four approaches in Table II are from our contributions, while the last two are based on [10] and [24].

TABLE II
Compared Algorithms

Notation	Failures	Disjointness	Embedding
S-CoViNE	Single	Algorithm 1	Algorithm 2
D-CoViNE	Double	Algorithm 1	Algorithm 2
S-CoViNE-ILP	Single	Algorithm 1	§ VI
D-CoViNE-ILP	Double	Algorithm 1	§ VI
S-Cutset-ILP [10]	Single	Optimal Cut-set	ILP
ViNE-ILP [24]	None	None	MCUF ILP[1]

1. Multi-commodity Unsplittable Flow

B. Simulation Setup

We implement the ILP formulations using IBM ILOG CPLEX C++ library. The simulations were performed on a server with quad-core 3.4GHz processor and 8GB of RAM. To demonstrate the scalability of our solutions, we consider both small and large network topologies as summarized in Table III. For each problem instance in this table, we perform 3 simulation runs and take the average. VNs for the small scale scenario are 2-edge connected, since it is required by the cut-set based approach [10]. We also vary the Link-to-Node Ratio (LNR) to assess the robustness of our solution for different VN connectivity levels. In addition to scalability and robustness, we analyze the behavior of our approach under different failure scenarios. Since our focus is VN connectivity, we use enough bandwidth capacity in SN topologies.

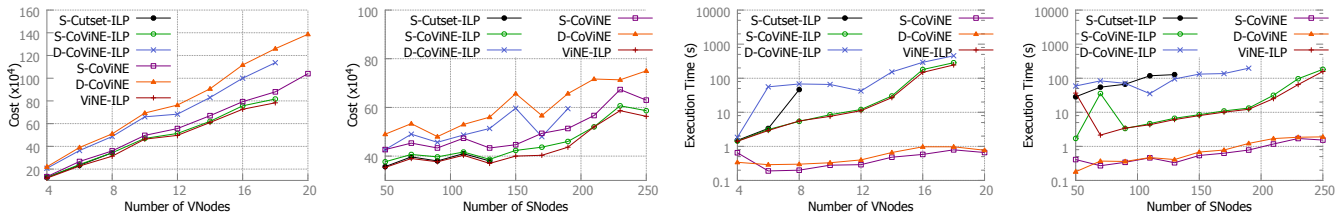
TABLE III
Summary of Simulation Parameters

Scenario	Figure	SNodes	SLinks	VNodes	VLinks
Small Scale	Fig. 3(a) Fig. 3(c)	150	310	4-20	5-37
	Fig. 3(b) Fig. 3(d)	50-250	105-494	10	17-24
Large Scale	Fig. 4(a)	500	2017	10-100	21-285
	Fig. 4(d)	1000	4023	10-100	21-285
	Fig. 4(b)	500	2017	10	11-31
		1000	4023	10	11-31
	Fig. 4(c)	500	1000-2000	10	21
		1000	2000-4000	10	21
Failure	Fig. 5	150	310	10	11-31

C. Results

1) Small Scale Scenarios:

a) Embedding Cost: Fig. 3(a) and Fig. 3(b) depict embedding cost for different VN and SN sizes, respectively. As expected, *ViNE-ILP* produces the lowest cost embedding, since it neither augments any parallel VLinks nor satisfies any disjointness constraint. The costs of embedding produced by *S-Cutset-ILP* and *S-CoViNE-ILP* lie very close to that of *ViNE-ILP*. Since the VNs are 2-edge connected in this experiment,



(a) Cost Vs. VN Size (b) Cost Vs. SN Size (c) Time Vs. VN Size (d) Time Vs. SN Size

Fig. 3. Small Scale Performance

no parallel augmentation is performed. The difference in *S-Cutset-ILP* and *S-CoViNE-ILP* is only due the variation of the disjointness computation methods. In contrast, *S-CoViNE* employs heuristic algorithms for disjointness computation and embedding, resulting in $\sim$10% more cost than *S-CoViNE-ILP* and $\sim$15% more cost than the cut set based optimal solution (*S-Cutset-ILP*). Both *D-CoViNE-ILP* and *D-CoViNE* incur $\sim$30% more cost than single failure approaches, since they augment parallel VLinks to ensure 3-edge connectivity. In general, cost increases almost linearly with increase in SN size for a fixed VN, and vice versa.

b) Execution Time: Fig. 3(c) and Fig. 3(d) present execution times in logarithmic scale by varying VN and SN sizes, respectively. Execution times of *S-CoViNE* and *D-CoViNE* vary almost linearly with VN or SN sizes. When embedding a VN of 18 VNodes on a 150 node SN, *S-CoViNE-ILP* and *D-CoViNE-ILP* take $\sim$ 285s and $\sim$ 456s, respectively, which is significantly slower compared to less than a second execution time for *S-CoViNE* and *D-CoViNE*. *ViNE-ILP* runs faster than *S-CoViNE-ILP* and *D-CoViNE-ILP*, since it does not satisfy any disjointness constraint while embedding. *S-Cutset-ILP* is the slowest since it computes an optimal solution.

c) Scalability: *S-CoViNE* and *D-CoViNE* can scale with arbitrary VN and SN sizes, whereas the ILP-based approaches can only scale up to 18 node VNs on 150 node SNs. Scalability of *S-Cutset-ILP* is the worst as it cannot scale beyond 10 node VNs on the same 150 node SNs. In summary, the higher costs of *S-CoViNE* and *D-CoViNE*, compared to the corresponding ILP-based approaches, are compensated by their higher scalability and faster execution time.

2) Large Scale Scenarios:

a) Embedding Cost: Fig. 4(a) shows embedding cost by varying VN sizes on SNs of 500 and 1000 nodes. On the other hand, Fig. 4(b) and Fig. 4(c) show embedding cost for VN and SN topologies with different LNRs, respectively. In this scenario, embedding cost is mostly influenced by disjointness constraint and parallel VLink augmentation. For double failure scenarios, augmentation cost dominates for VN LNR ≤ 2.4, hence the initial decrease in embedding cost. However for VN LNR > 2.4, cost for ensuring disjointness constraint dominates, which justifies the corresponding increase in Fig. 4(b). On the other hand for *S-CoViNE*, disjointness constraint dominates and embedding cost increases as higher number of VLinks are embedded on the same SN for larger LNR. An increase in SN LNR results into higher path diversity in SN. *S-CoViNE* and *D-CoViNE* exploit this path diversity by finding shorter paths while embedding a VLink. This accounts for the decrease in cost with an increase in SN LNR.

b) Execution Time: Conforming to the running time analysis in § V and § VII, the execution times for *S-CoViNE* and *D-CoViNE* increase with both VN and SN sizes (Fig. 4(d)).

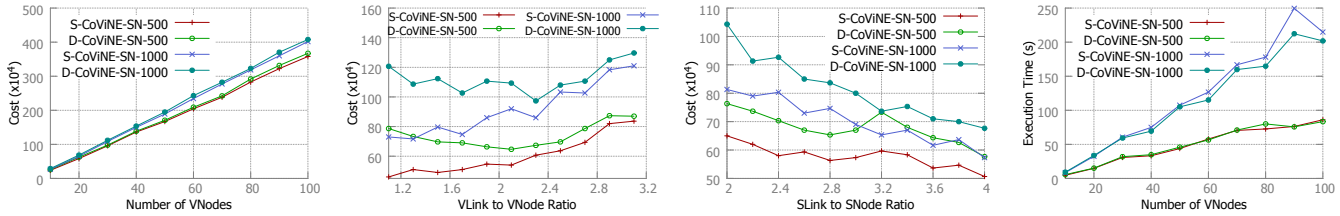
3) Impact of Failure: In this scenario, we assume three traffic classes in VN, labeled as 1 (highest priority), 2 and 3 (lowest priority) demanding 20%, 30%, and 50% of each VLink's bandwidth, respectively. We handle failures by rerouting traffic in the affected VLinks along alternate shortest paths in VN. Bandwidth sharing along these paths follow fair sharing policy between traffic from the same class and weighted fair sharing across different traffic classes.

Fig. 5(a) and Fig. 5(b) present the percentage of restored bandwidth for single and double failure scenarios, respectively. On the other hand, Fig. 5(c) and Fig. 5(d) present the overhead for ensuring connectivity in terms of embedding cost and number of augmented VLinks, respectively. Fig. 5(a) and Fig. 5(b) depict that performance of our embedding heuristics (*S-CoViNE* and *D-CoViNE*) is very close to the optimal embedding (*S-CoViNE-ILP* and *D-CoViNE-ILP*) for all three traffic classes. The percentage of restored bandwidth by ViNE-ILP is very poor at low VN LNR and increases with the increase in VN LNR. A higher LNR induces higher path diversity in VN. This has twofold impact. First, it reduces the chances of VN partitioning. Second, there are more options for steering traffic in the affected VLinks. Both of these reasons contribute to the increase in restored bandwidth for VN with higher LNR.

As envisioned at the beginning of this paper, our approach is able to successfully restore almost the full bandwidth for the highest priority traffic in presence of single and double failures as shown in Fig. 5(a) and Fig. 5(b). However, this successful restoration is at the expense of penalizing the lower priority traffic classes. The overall decrease in restored bandwidth for all variants of *CoViNE* with increasing VN LNR is counter-intuitive. This can be explained by observing the overheads in Fig. 5(c) and Fig. 5(d). As VN LNR increases, the number of augmented VLinks decreases. This results into lower spare bandwidth in VNs with higher LNR, and consequently reducing the percentage of restored bandwidth.

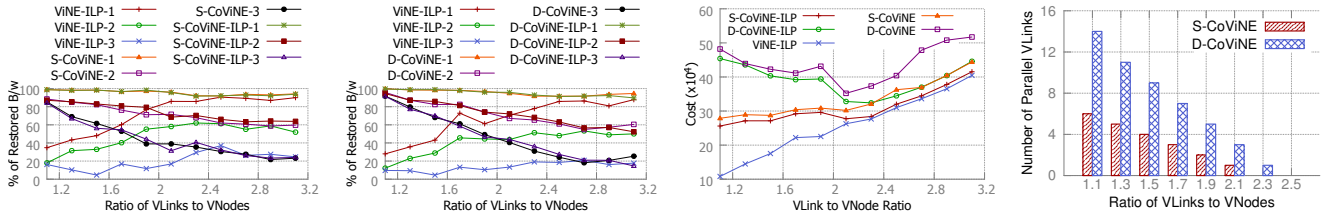
IX. Conclusion

In this paper, we have investigated the **Co***nnectivty-aware* **Vi***rtual* **N***etwork* **E***mbedding* (CoViNE) problem that ensures VN connectivity in presence of multiple SLink failures. We



(a) Cost Vs. VN Size (b) Cost Vs. VN LNR (c) Cost Vs. SN LNR (d) Time Vs. VN Size

Fig. 4. Large Scale Performance



(a) Single Failure (b) Double Failure (c) Overhead (d) Parallel VLink Augmention

Fig. 5. Impact of Failure

have addressed the two major challenges in solving *CoViNE*: i) finding the conflicting VLinks that should be embedded disjointly, and ii) computing a resource efficient embedding that adheres to disjointness requirement. For the first challenge, we have coined the concept of conflicting set, and have proven that computing optimal conflicting set is NP-complete. We also provided a heuristic algorithm for finding conflicting set efficiently. For the second challenge, we provided an ILP formulation and a heuristic to tackle its computational complexity. All of our solutions are generalized to handle multiple SLink failures for any VN and SN topology. Evaluation results show that solutions from our heuristics use around 15% extra resources on average compared to the optimal solution, whereas the execution time of our heuristic is two to three orders of magnitude faster on the same problem instances. We have also demonstrated that VN connectivity can be successfully applied to restore high priority traffic in presence of multiple SLink failures.

We believe that *CoViNE* can set the stage for further research investigations. Among the possibilities, we want to investigate the problem of ensuring different connectivity levels for each VLink in a VN, which can empower a VN-operator to offer a wide variety of Service Level Agreements (SLAs) to its customers. We also want to extend our current solutions by considering SLinks' spare bandwidth allocation, SNodes' throughput, and substrate path length constraints in a coordinated manner.

Acknowledgment

This work was supported in part by Huawei Technologies and in part by an NSERC Collaborative Research and Development Grant.

References

[1] N. M. M. K. Chowdhury *et al.*, "A Survey of Network Virtualization," *Computer Networks*, Apr 2010.

[2] M. R. Rahman *et al.*, "SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization," *IEEE TNSM*, 2013.

[3] A. Markopoulou *et al.*, "Characterization of Failures in an IP Backbone," in *INFOCOM*, Mar 2004.

[4] S. Herker *et al.*, "Survey on Survivable Virtual Network Embedding Problem and Solutions," in *ICNS*, 2013.

[5] Z. Zhou *et al.*, "Cross-layer network survivability under multiple cross-layer metrics," *IEEE/OSA J. of Optical Comm. & Net.*, 2015.

[6] H. Choi *et al.*, "Loopback recovery from double-link failures in optical mesh networks," *IEEE/ACM TON*, Dec 2004.

[7] P. Gill *et al.*, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *ACM SIGCOMM*, Aug 2011.

[8] K. Thulasiraman *et al.*, "Logical topology augmentation for guaranteed survivability under multiple failures in ip-over-wdm optical networks," *Optical Switching and Networking*, vol. 7, no. 4, pp. 206–214, 2010.

[9] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," Ph.D. dissertation, Citeseer, 1996.

[10] E. Modiano *et al.*, "Survivable lightpath routing: a new approach to the design of wdm-based networks," *IEEE JSAC*, 2002.

[11] A. Todimala *et al.*, "A scalable approach for survivable virtual topology routing in optical wdm networks," *IEEE JSAC*, 2007.

[12] K. Thulasiraman *et al.*, "Circuits/cutsets duality and a unified algorithmic framework for survivable logical topology design in ip-over-wdm optical networks," in *IEEE INFOCOM*, Apr 2009.

[13] Z. Zhou *et al.*, "Novel survivable logical topology routing in ip-over-wdm networks by logical protecting spanning tree set," in *ICUMT*, 2012.

[14] M. Kurant *et al.*, "Survivable mapping algorithm by ring trimming (smart) for large ip-over-wdm networks," in *BroadNets*, Oct 2004.

[15] T. Guo *et al.*, "Shared backup network provision for virtual network embedding," in *IEEE ICC*, 2011.

[16] J. Xu *et al.*, "Survivable virtual infrastructure mapping in virtualized data centers," in *IEEE CLOUD*, 2012.

[17] M. M. A. Khan *et al.*, "Simple: Survivability in multi-path link embedding," in *IEEE CNSM*, 2015, pp. 210–218.

[18] K. Lee *et al.*, "Cross-layer survivability in wdm-based networks," *IEEE/ACM TON*, Aug 2011.

[19] C. Liu *et al.*, "A new survivable mapping problem in ip-over-wdm networks," *IEEE JSAC*, Apr 2007.

[20] M. Kurant *et al.*, "Survivable routing of mesh topologies in ip-over-wdm networks by recursive graph contraction," *JSAC*, 2007.

[21] D. D.-J. Kan *et al.*, "Lightpath routing and capacity assignment for survivable ip-over-wdm networks," in *IEEE DRCN*, 2009.

[22] Menger's theorem [online] http://math.fau.edu/locke/menger.htm.

[23] M. Melo *et al.*, "Virtual network mapping–an optimization problem," in *Mobile Networks and Management*. Springer, 2012, pp. 187–200.

[24] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *IEEE INFOCOM*, 2006.

Time-Aware Congestion-Free Routing Reconfiguration

Shih-Hao Tseng, Chiun Lin Lim, Ning Wu, and Ao Tang
School of Electrical and Computer Engineering, Cornell University, Ithaca, New York 14853, U.S.A.
Email: {st688, cl377, nw276}@cornell.edu, atang@ece.cornell.edu

***Abstract*—A general model is developed to study how network routing can be reconfigured quickly without incurring transient congestion. Assuming both initial and target configurations are congestion-free, it is known that transient congestion may still occur during the reconfiguration process if links contain a mix of traffic flows following old and new routing rules, resulting from variation of switch reaction time and propagation delay differences among paths. We consider these factors by explicitly incorporating timing uncertainty intervals into the model. The model leads to an optimization problem whose solution represents a fast (in terms of actual physical time) congestion-free routing reconfiguration. Our formulation naturally reduces to existing work of finding minimal number of algorithmic update steps when the timing uncertainty intervals are very large, meaning we have little prior knowledge about them. The optimization problem is shown to be a Mixed Integer Linear Program (MILP) with a polynomial-size constraint set, and is proved to be NP-hard. We then further introduce an approximation algorithm with performance guarantee to solve the problem efficiently. Several numerical examples are provided to illustrate our results. In particular, it is demonstrated that timing information can possibly accelerate the update process, even if more steps are involved.**

I. Introduction

Network routes are frequently reconfigured to accomplish tasks such as middlebox traversal constraint satisfaction, virtual machine live migration, and scheduled network maintenance [1]–[5]. There are three key challenges for route reconfiguration, namely *optimality*, *consistency* and *swiftness*.

An optimal update ensures the final routing configuration is the targeted optimal solution. Network operators derive the optimal solution from the new network conditions and traffic demands by solving the well-known multi-commodity flow problem [6]–[8]. In practice, we have various protocols attempting to achieve the *static* solution of the multi-commodity flow problem. Under a distributed setting, routes could be reconfigured with switches choosing different per-destination next-hops as in link-state routing protocols such as OSPF [9]. Alternatively, the routes could be predetermined and route reconfiguration could be done by ingress switches individually selecting different tunnels as in MPLS [10]. A drawback of these distributed methods is that the achieved configuration is not necessarily the desired one [11], [12]. However, having a centralized controller with a global view, as in a Software Defined Network (SDN), can guarantee optimality by directly establishing the optimal routing configuration.

With a centralized controller guaranteeing optimality, the concern moves on to the *transient* stages while getting to the optimal solution, and this is where consistency and swiftness requirements come in. A consistent update ensures certain network properties of interest, such as in-order delivery, loop-freedom or capacity constraint, are satisfied during all transient stages of the routing reconfiguration [13]–[16]. When implemented incorrectly, an inconsistent update could be a very costly exercise to the operator by causing severe service disruptions that would take days to fully recover [17]. On the other hand, swiftness refers to the ability to reconfigure the network from the initial state to the target state in the least amount of time possible. A swift update prevents the new routing setup from becoming obsolete due to fast changing network conditions. This becomes increasingly critical especially for data center networks where traffic dynamics fluctuates very fast [18].

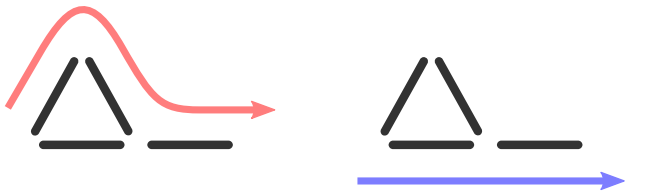
Recently, several methods have been developed to acquire timing information in the network [19]–[21], which enable us to present our approach to achieve fast congestion-free routing reconfiguration. Given an initial routing configuration and a target one, our goal is to produce a series of update steps to move from the initial to the target configuration as quickly as possible while congesting no link during any update step. The key problem is that congestion can still occur during the transient even if both old and new routing configurations are congestion-free, since traffic flows following the new configuration could enter the links containing some traffic flows keeping the old one. We capture this behavior by explicitly incorporating timing information such as propagation delay and time variability into updates.

Our work differs from prior works [15], [22], [23] by optimizing reconfiguration time instead of the number of algorithmic update steps. The key motivation behind the extension is that minimizing the number of update steps does not necessarily translate to a faster update (Example 3). Timing information is useful for network operators to achieve faster reconfiguration as it helps rule out the impossible scenarios which are still considered by worst-case analysis [15]. Our framework reduces to SWAN [15] when the uncertainty dominates the network and we have essentially no prior timing knowledge. zUpdate [22] is also a special case of our framework when we have perfect timing information (zero uncertainty) and the network has layered structure.

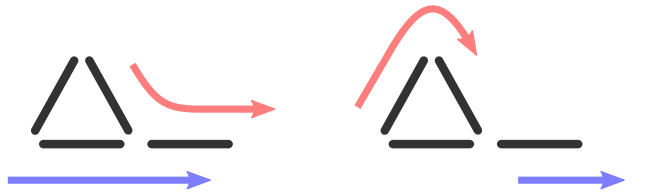
II. Background and Motivation

As discussed in Section I, even if initial and the target configurations both obey capacity constraints, congestion may still occur during transient stages. There are two main factors that can lead to transient congestion: propagation delay and timing uncertainty. In this section, we provide two examples

ISBN 978-3-901882-83-8

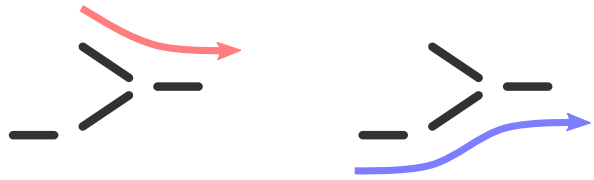


(a) The initial configuration (b) The target configuration

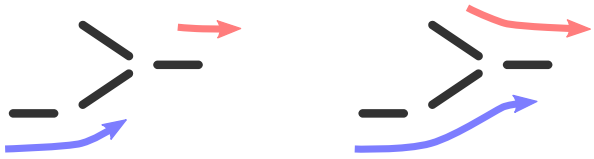


(c) Transient congestion (d) A congestion-free update exists when the bottom flow switches to the top path

Fig. 1. Differences in propagation delay can cause congestion



(a) The initial configuration (b) The target configuration



(c) A congestion-free update (d) Uncertainty causes congestion

Fig. 2. Timing uncertainty can cause congestion

(Example 1, 2) to intuitively illustrate how that can happen. Furthermore, we also demonstrate how timing information can help achieve faster reconfiguration (Example 3).

A. Causes of Transient Congestion

1) Propagation Delay: When switching traffics between different paths, the differences between propagation delays of the paths may incur transient congestion. We will illustrate this phenomenon with the following example.

Example 1. Consider the network as shown in Fig. 1. The flow along the top path (Fig. 1(a)) is directly rerouted to the bottom one (Fig. 1(b)). Assume the bottom path has a shorter delay, the new flow can arrive at the rightmost link before the previous flow on the longer path clears there (Fig. 1(c)). The shared link of the two paths may thus exceed its capacity and cause congestion.

Propagation delay can cause congestion as Example 1 reveals. However, it also grants better performance. We will discuss this opportunity in Section II-B with Fig. 1(d).

2) Timing Uncertainty: In practice, network operator cannot precisely specify when each path reconfiguration is executed. This can be due to imperfect synchronization among different switches. Some other reasons include the inevitable varying reaction time of different switches for an update instruction [23] as well as varying processing time that cannot be accurately predicted [24].

Although accurate prediction is practically hard, estimations still help order the upcoming events. The arrivals and the clearances of flows on a link can be depicted as intervals, and the uncertainty is reflected by the length of the interval, which converges to zero if the event timing is certain. The overlapping intervals indicate that the link may consist of flows in different configurations at the same time. In the extreme case, all the intervals intersect with each other because of uncertainty, and we call this scenario *order-oblivious*.

Example 2. Fig. 2 shows a part of a network. The operator shifts the flow along the top path (Fig. 2(a)) to somewhere else and routes another flow to the bottom path (Fig. 2(b)). If the two source switches perform the update without uncertainties, the reconfiguration is congestion-free (Fig. 2(c)). However, uncertainty of the source switch on the top may postpone the update and congest the rightmost link (Fig. 2(d)).

B. Benefits of Timing Information

Previous work considers only the order-oblivious worst-case scenario, which assumes the arrivals of the new flows are independent of the others [15]. However, better performance is possible as the timing information reveals the arriving order.

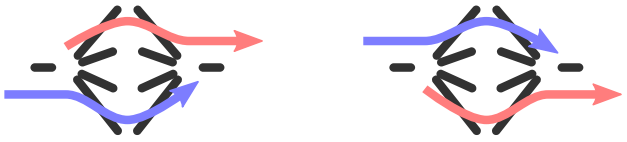
To illustrate this, reversed update in Fig. 1 is considered: moving the flow from the bottom path to the top one. The clearance of the old flow finishes earlier than the arrival of the new flow on the rightmost link (Fig. 1(d)), which suggests a congestion-free one-shot update. Nevertheless, order-oblivious scenario rejects this possibility if the shared link cannot accommodate the two flows simultaneously.

In addition to enlarging the set of feasible update sequences, timing information in general can help accelerate the reconfiguration process. We define the *required time* as the minimum time needed to ensure the change has been fully deployed to the network. For example, the required time for a flow variation along a path is the maximum time needed for the new flow to propagate through the path; the required time for the clearance of an update step is the maximum required time for the flows varied in the step (is zero if all the flows stay the same); the required time for an update sequence is the sum of the required time for the clearance of each step.

Adopting required time can expedite an update. Previous work finds the least step update sequence without timing information [15]. At each step, a long fixed period of time should be waited to avoid the interference between consecutive steps. Replacing the fixed time with the required time yields a more aggressive update. Moreover, an update can achieve shorter required time with more steps.

TABLE I
RELEVANT CONFIGURATIONS

Configurations	Top Traffic				Bottom Traffic			
Initial	0	2	0	0	0	0	2	0
Target	0	0	2	0	0	2	0	0
Intermediate 1	1	0	0	1	0	0	0	2
Intermediate 2	1	1	0	0	0	1	1	0
Intermediate 3	1	0	1	0	0	2	0	0



(a) The initial configuration (b) The target configuration

Fig. 3. Minimizing the number of update steps does not necessarily minimize the update time

Example 3. In Fig. 3, two traffics with the same rate 2 are swapped in a network without uncertainty. Each traffic has four candidate acyclic paths to flow through, and the capacities of the paths are $1, 2, 2, 4$ respectively from the top to the bottom. Each link causes propagation delay 1 unit time except for the bottom two 5-unit delay links. Table I lists the relevant configurations. The columns under each traffic are the flows on the paths. The leftmost column corresponds to the topmost path, and the rightmost column refers to the bottommost one.

Clearly, one-shot update is infeasible, and thus we need to stagger the update into multiple steps. Least step update involves two steps by using the bottom spare path (Intermediate 1). The required time for the update is $11 + 11 = 22$.

However, we can leverage the top path and shift half top flow through it first. The bottom flow is also split half to its target path (Intermediate 2). In the second step, we shift the flows to the target except for the unit flow along the top path (Intermediate 3). Clearing the flow on the top path serves as the last step of the three-step update, which has the required time $3 + 3 + 3 = 9$ faster than the least step solution.

III. FORMULATION

Section II discloses the potential of timing information to achieve a fast congestion-free reconfiguration. Similar to [15], we are interested in congestion-free update in upper bounded number of steps. **In contrast to finding the least-step solution as in [15], we seek the fastest solution.** In this section, we introduce the notations to model the centralized routing scheme and express our model as an optimization problem in Section III-B3.

A. Notations

A centralized controller controls a network consisting of a set V of switches and a set L of directed links. N users utilize the network, and each user n demands a traffic rate d^n from a source switch to a destination switch (or simply the source/destination). A set of acyclic paths P^n is predetermined and established for each user n to communicate between the source and the destination. The controller performs routing by specifying the split ratio among the paths in P^n for each user n at its source. Define $P = \bigcup_{n \in N} P^n$ as all acyclic paths.

TABLE II
DEFINITIONS OF THE MAJOR VARIABLES

For each link $l \in L$	
c_l	The capacity on the link l
f_l	The total flow on the link l
P_l	The set of paths passing through the link l
$P_l^{\mp}$	A P-partition (P_l^-, P_l^+)
$\mathcal{P}_l$	The collection of all valid P-partitions of P_l
For each user $n \in N$	
d^n	The demanded flow rate into the network for the user n
P^n	The set of acyclic paths for the user n
For each path $p \in P$	
x_p	The flow on the path p
z_p	The binary integer variable for the path p
$[w_p^{\min}, w_p^{\max}]$	The time interval for the updated flow to clear the path or subpath p
For each level $0 \le \pi \le \pi^{\max}$ (see Section IV)	
r_π	The level variable for the level π
$\overline{w_\pi^{\max}}$	The required time for the level π
Other variable	
u	The required time to finish an update step

For each link l in a path $p \in P$, q_{pl} denotes the subpath from the source to the switch prior to link l. For example, if $p = (v_1, l_1, v_2, l_2, v_3)$ connects the source v_1 and the destination v_3, $q_{pl_2} = (v_1, l_1, v_2)$. Define $P_l = \{p \in P : l \in p\}$ as the paths passing through link l, which is a subset of P.

Define x_p as the rate of the *flow* along a path p. f_l denotes the total flow injecting to link l. Due to the timing issues discussed in Section II, we assume the delays to pass through switches and links are interval-ranged. Those delays are accumulated along the paths and encountered by the flows. An interval-ranged delay is also assumed for an update instruction between being issued and becoming effective.

The interval $[w_p^{\min}, w_p^{\max}]$ specifies the time needed for a packet to go through a path p. The lower bound $w_p^{\min}$ is derived by adding all the lower bound of the delays for the source-switch instruction update and the on-path device transitions. Analogous computation gives the upper bound $w_p^{\max}$. Notice that $w_p^{\max}$ is the required time for the path p, which is introduced in Section II-B. We can define the interval $[w_q^{\min}, w_q^{\max}]$ for a subpath q in the similar way.

An update sequence generally consists of multiple steps, which are labeled by step numbers a in chronological order. The parentheses enclosing a is attached behind a variable to denote the value of the variable between step a and $a + 1$. For example, $x_p(a)$ is the flow along path p after the network applies update step a; $u(a)$ is the required time for the update to fully switch from step a to $a+1$ configuration. t_a refers to the time elapsed since the step a is applied to the network.

We may drop the subscript of path p or the step number a to refer to a vector, such as x and u.

B. Model

1) Objective: Given an upper bound b, our objective is to find the fastest update sequence in b steps to reconfigure the network from a given initial state to a given target state, both static feasible and congestion-free, while remaining

congestion-free during the whole transient stage. The initial and the target routing configurations are denoted by $x_p(0)$ and $x_p(b)$ for all $p \in P$, respectively. Without loss of generality, we assume there exists $p \in P$ such that $x_p(0) \neq x_p(b)$. Finding an update sequence in b steps involves deciding a series of flows $x_p(a)$ for all $p \in P$ and $a = 1, \ldots, b-1$.

For simplicity, we impose the assumption that the user traffic demands remain the same during the whole update, although our framework can easily deal with general cases. Extending the update mechanism in [15], we also assume the centralized controller waits the required time $u(a)$ between consecutive steps a and $a+1$ to ensure the full deployment of the new update $(a+1)$ before performing the next $(a+2)$.

2) Constraints: According to the update mechanism assumption in Section III-B1, each flow can only follow either step a or $a+1$ configuration at an arbitrary time t_a after each step a. Congestion-free property requires that all the possible combinations of the flows at each link do not exceed the capacity. The number of potential combinations is exponential to the number of flows [22], thus it is critical to identify the essential ones. Example 4 shows how timing information helps select the relevant constraints and avoid enumerating all the unnecessary combinations.

Example 4. Consider a link l shared by two paths $p_1, p_2 \in P$. For simplicity, we use $q_1 = q_{p_1 l}$ and $q_2 = q_{p_2 l}$ in this example.

Once updated from step a to $a+1$, the flow along p_i arrives at the link l within the time interval $\left[w_{q_i}^{\min}, w_{q_i}^{\max}\right]$ for $i = 1, 2$. When the time intervals overlap, such as $w_{q_1}^{\min} \leq w_{q_2}^{\min} \leq w_{q_1}^{\max}$, either flow can arrive before the other. That results in the order-oblivious scenario, and all possible combinations should be covered by the following four constraints:

$$\begin{aligned} x_{p_1}(a) + x_{p_2}(a) &\leq c_l, \quad & x_{p_1}(a+1) + x_{p_2}(a) &\leq c_l, \\ x_{p_1}(a) + x_{p_2}(a+1) &\leq c_l, \quad & x_{p_1}(a+1) + x_{p_2}(a+1) &\leq c_l, \end{aligned}$$

or we can represent them as

$$\max(x_{p_1}(a), x_{p_1}(a+1)) + \max(x_{p_2}(a), x_{p_2}(a+1)) \leq c_l.$$

When the two intervals are separated, assuming $w_{q_1}^{\max} < w_{q_2}^{\min}$ without loss of generality, $x_{p_1}(a+1)$ always arrives earlier than $x_{p_2}(a+1)$. Instead of considering all four combinations of flows, only three combinations are possible and they lead to the less strict constraint set:

$$\begin{aligned} x_{p_1}(a) &+ x_{p_2}(a) &&\leq c_l, \\ x_{p_1}(a+1) &+ \max(x_{p_2}(a), x_{p_2}(a+1)) &&\leq c_l. \end{aligned}$$

To write down the congestion-free constraints, we define a *P-partition* of P_l as a pair of subsets $P_l^{\mp} = (P_l^-, P_l^+)$ such that $P_l^- \cap P_l^+ = \emptyset$ and $P_l^- \cup P_l^+ = P_l$. With the P-partition notation, we can express each of the $2^{|P_l|}$ order-oblivious constraints at each link l in the following form:

$$f_l(P_l^{\mp}, a) = \sum_{p \in P_l^-} x_p(a) + \sum_{p \in P_l^+} x_p(a+1) \leq c_l$$

where $f_l(P_l^{\mp}, a)$ is the corresponding total flow on the link. In Example 4, $P_l = \{p_1, p_2\}$, and four possible P-partitions $P_l^{\mp} = (P_l^-, P_l^+)$ are $(\{p_1, p_2\}, \emptyset)$, $(\{p_1\}, \{p_2\})$, $(\{p_2\}, \{p_1\})$, and $(\emptyset, \{p_1, p_2\})$, corresponding to the four $f_l(P_l^{\mp}, a) \leq c_l$ constraints respectively.

We then characterize the P-partitions $P_l^{\mp}$ which lead to possible flow combinations $f_l(P_l^{\mp}, a)$. A P-partition is *valid* if there exists a time interval $[t_a, t_a + \epsilon]$ for some constant $\epsilon > 0$[1], such that all the flows in P_l^+ have its update propagated to link l while those in P_l^- still remain in the old configuration. We define the collection $\mathcal{P}_l$ as the set of all the valid P-partitions of P_l. As demonstrated in Example 4, we can decide the validity of a P-partition by scrutinizing the arrival intervals of the flows. In particular, there must exist $t_a \geq 0$ and $\epsilon > 0$ for a valid P-partition such that $t_a \geq w_{q_{pl}}^{\min}$ for every $p \in P_l^+$ and $t_a + \epsilon \leq w_{q_{pl}}^{\max}$, for every $p \in P_l^-$. Therefore, we have the following equation:

$$\max_{q_{pl}: p \in P_l^+} w_{q_{pl}}^{\min} < \min_{q_{pl}: p \in P_l^-} w_{q_{pl}}^{\max}. \tag{1}$$

Since $|P|$ is finite, both sides of the inequality are attained finite, which implies the existence of ϵ by setting it as the gap. It is clear that a P-partition satisfies the condition (1) if and only if it is valid.

3) Optimization Problem: Now we summarize the objective in Section III-B1 and the constraints in Section III-B2 into an optimization problem. Our objective can be expressed as an optimization problem of minimizing the objective function $\sum_{a=0}^{b-1} u(a)$, which is the required time for the resulted update sequence. At each step a, the required time can be written as

$$u(a) = \max\left\{w_p^{\max} : x_p(a+1) \neq x_p(a)\right\}.$$

By convention, we set the expression to zero if none of the flows is updated. We create the artificial binary integer variable $z_p(a)$ for each path p, which is one if and only if $x_p(a+1) \neq x_p(a)$, so that

$$u(a) = \max_{p \in P}\left\{w_p^{\max} z_p(a)\right\}. \tag{2}$$

We name the optimization problem Fast Congestion-free Reconfiguration problem in b steps (FCR(b), or simply FCR for an arbitrary b), and formulate it in a form of Mixed Integer Linear Programming (MILP):

$$\begin{aligned} \min \quad & \sum_{a=0}^{b-1} u(a) \\ \text{s.t.} \quad & f_l(P_l^{\mp}, a) \leq c_l && \forall l \in L, P_l^{\mp} \in \mathcal{P}_l \\ & && 0 \leq a \leq b-1 && (3) \\ & \sum_{p \in P^n} x_p(a) = d^n && \forall n \in N, 1 \leq a \leq b-1 && (4) \\ & x_p(a) \geq 0 && \forall p \in P, 1 \leq a \leq b-1 && (5) \\ & x_p(0), x_p(b) \text{ are given} && \forall p \in P && (6) \\ & u(a) \geq w_p^{\max} \cdot z_p(a) && \forall p \in P, 0 \leq a \leq b-1 && (7) \\ & z_p(a) \cdot \alpha_p \geq |x_p(a+1) - x_p(a)| && \forall p \in P, 0 \leq a \leq b-1 && (8) \\ & z_p(a) \in \{0, 1\} && \forall p \in P, 0 \leq a \leq b-1 && (9) \end{aligned}$$

[1]By requiring $\epsilon > 0$, we exclude the case that the network congests for 0 time.

where the constant α_p for path p is set as the bottleneck link capacity $\min_{l\in p} c_l$ so that $z_p(a)$ behaves as expected.

The constraints (3) - (6) are the feasibility constraints. x satisfying these constraints gives a feasible congestion-free update sequence. We introduce the constraints (7) - (9) to realize the relationship (2). The optimal value of FCR(b) is written as $\mathrm{OPT}_{\mathrm{FCR}}(b)$ for given step upper bound b. One important observation is that if FCR is feasible in b steps, it is also feasible for $b+1$ steps and $\mathrm{OPT}_{\mathrm{FCR}}(b+1) \leq \mathrm{OPT}_{\mathrm{FCR}}(b)$. Conversely, if the problem is not feasible in b steps, there exists no solution for fewer steps.

IV. Main Results

In Section III-B, we formulate the reconfiguration problem as an MILP problem FCR. In this section, we present a proof sketch of the NP-hardness of FCR, which motivates us to develop an approximation algorithm for FCR. In particular, we provide a polynomial time relaxation-rounding based approximation algorithm, which involves only the Linear Program (LP) solutions. We start the section with Theorem 1.

Theorem 1. FCR *is NP-hard.*

Theorem 1 can be proved by showing that 3-SAT, a well-know NP-complete problem, polynomial reduces to FCR. For each 3-SAT instance, there exist a network and its corresponding initial and target configurations such that the optimal FCR solution meets the lower bound if and only if the 3-SAT instance is satisfiable. We omit the details as they are not the main focus of this work.

Since FCR is NP-hard, we develop an approximation algorithm for FCR as the following. We first define the term *kernel*. A kernel $A(b)$ is an algorithm which gives a feasible solution to FCR(b) when it is feasible; and FCR is not feasible in b steps if $A(b)$ finds no solution to it. We write $\mathrm{SOL}_A(b)$ to denote the objective value of the feasible solution given by the kernel $A(b)$, and we set $\mathrm{SOL}_A(b) = \infty$ if the kernel $A(b)$ declares FCR(b) infeasible. We then propose the algorithm $\mathrm{ALG}[A](b)$ (Algorithm 1) with respect to a kernel $A(\cdot)$, which essentially picks the best solution from the feasible solutions generated by $A(1), A(2), \cdots, A(b)$.

Algorithm 1 Algorithm $\mathrm{ALG}[A](b)$

1: $\mathrm{SOL}_{\mathrm{ALG}[A]}(b) \leftarrow \min_{1\leq\hat{b}\leq b} \mathrm{SOL}_A(\hat{b})$.
2: **if** $\mathrm{SOL}_{\mathrm{ALG}[A]}(b) = \infty$ **then**
3: Output "no congestion-free solution in b steps."
4: **else**
5: Output the solution corresponding to the minimum $\mathrm{SOL}_A(\cdot)$.
6: **end if**

If the required time $u(a)$ for each step given by the kernel A is upper bounded by a constant $W^{\max}$ and the required time $w_p^{\max}$ for each path is lower bounded by a constant $W^{\min}$, we have the following theorem to ensure that $\mathrm{ALG}[A](b)$ is a $\frac{W^{\max}}{W^{\min}}$-approximation algorithm.

Theorem 2. *Let $W^{\max}$ and $W^{\min}$ be positive constants and u belong to the feasible solution to $\mathrm{FCR}(b)$ given by a kernel $A(b)$. $\mathrm{ALG}[A](b)$ is a $\frac{W^{\max}}{W^{\min}}$-approximation algorithm, if the following conditions hold:*

- $0 \leq u(a) \leq W^{\max}$ *for all* $0 \leq a \leq b-1$.
- $w_p^{\max} \geq W^{\min}$ *for all* $p \in P$.

Proof. If FCR(b) is infeasible, the kernel algorithm gives $\mathrm{SOL}_A(\hat{b}) = \infty$ for all $1 \leq \hat{b} \leq b$ and $\mathrm{ALG}[A](b)$ declares infeasibility of FCR(b). If FCR(b) is feasible, there exists $1 \leq b' \leq b$ such that $OPT_{FCR}(b') = OPT_{FCR}(b)$ and $u(a) > 0$ for all $0 \leq a \leq b'-1$. Since $u(a) \geq w_p^{\max} z_p(a)$ for all $p \in P$, we know $\exists p' \in P$ such that $z_{p'}(a) = 1$, and hence $u(a) \geq w_{p'}^{\max} \geq W^{\min}$. Denote this optimal u by u_{OPT}. Let the solution u corresponding to $\mathrm{SOL}_A(b)$ be u_b, we know

$$\mathrm{SOL}_{\mathrm{ALG}[A]}(b) = \min_{1\leq\hat{b}\leq b} \mathrm{SOL}_A(\hat{b})$$
$$\leq \mathrm{SOL}_A(b') = \sum_{a=0}^{b'-1} u_{b'}(a)$$
$$\leq \sum_{a=0}^{b'-1} W^{\max} \leq \sum_{a=0}^{b'-1} W^{\max} \frac{u_{\mathrm{OPT}}(a)}{W^{\min}} = \frac{W^{\max}}{W^{\min}} \sum_{a=0}^{b'-1} u_{\mathrm{OPT}}(a)$$
$$= \frac{W^{\max}}{W^{\min}} OPT_{FCR}(b') = \frac{W^{\max}}{W^{\min}} OPT_{FCR}(b).$$

□

There are generally more than one kernels satisfying the first condition in Theorem 2. For instance, we have the **kernel $R_{\mathrm{FCR}}(b)$**, which solves the linear relaxation of FCR(b), uprounds all the resulted non-zero $z_p(a)$ to 1 and sets $u(a) = \max_{p:z_p(a)=1} w_p^{\max} \leq \max_{p\in P} w_p^{\max}$. In that case, $W^{\max} = \max_{p\in P} w_p^{\max}$ and $\mathrm{ALG}[R_{\mathrm{FCR}}](b)$ is a $\frac{\max_{p\in P} w_p^{\max}}{\min_{p\in P} w_p^{\max}}$-approximation algorithm.

Unlike the kernel-independent approximation ratio, the complexity of $\mathrm{ALG}[A](b)$ depends on the complexity of its kernel. If the kernel $A(b)$ is a polynomial-time algorithm, $\mathrm{ALG}[A](b)$ also terminates in polynomial time. LP-based kernel, such as $R_{\mathrm{FCR}}(b)$, is indeed polynomial-time when the constraint set is also in polynomial size of its input variables. Among the constraints of FCR, we only need to find an polynomial-size expression of the constraint (3) to have a polynomial-time LP-based kernel. Equivalently, we need to consider all valid P-partitions $P_l^{\mp}$ and their corresponding total flows $f_l(P_l^{\mp}, a)$. We develop a polynomial time constraint set generation algorithm (Algorithm 2) and give a theorem (Theorem 3) to prove that the generated constraint set is equivalent to the one derived from all valid P-partitions.

Theorem 3. *A solution x satisfies the constraint (3) if and only if there exists s such that (x, s) is feasible for the constraints generated by the Algorithm 2.*

Theorem 3 is proved by showing that every valid P-partition can be expressed as $(\Sigma_y \cup \Sigma_c^-, \Sigma_c^+ \cup \Sigma_u)$ and each $(\Sigma_y \cup \Sigma_c^-, \Sigma_c^+ \cup \Sigma_u)$ is a valid P-partition, where Σ_y, Σ_c and Σ_u are obtained at some iteration of Algorithm 2 and Σ_c^-, Σ_c^+ is a partition of Σ_c. The proof is straightforward, and we omit the details here because of the space limitation.

Besides the size of the constraint set, the number of input variables also plays an important role in the time complexity

Algorithm 2 Constraint Set Generation

for each path $p \in P$ and $0 \le a \le b-1$ **do**
  Add a slack variable $s_p(a)$ and two slack constraints
$$s_p(a) \ge x_p(a), \qquad s_p(a) \ge x_p(a+1).$$
end for
for $l \in L$ and $a = 0$ **to** $b-1$ **do**
  $P_l \leftarrow \{p \in P : l \in p\}$
  Set the updated set $\Sigma_u \leftarrow \emptyset$.
  Set the current set $\Sigma_c \leftarrow \emptyset$.
  Set the yet-updated set $\Sigma_y \leftarrow P_l$.
  Collect $w^{\max}_{q_{pl}}$ and $w^{\min}_{q_{pl}}$ for all $p \in P_l$ to be a list.
  for w in the list from the smallest to the largest **do**
    while $w = w^{\min}_{q_{pl}}$ for some $p \in \Sigma_y$ **do**
      Remove p from Σ_y and add it to Σ_c.
    end while
    while $w = w^{\max}_{q_{pl}}$ for some $p \in \Sigma_c$ **do**
      Remove p from Σ_c and add it to Σ_u.
    end while
    Generate a constraint
$$\sum_{p\in\Sigma_y} x_p(a) + \sum_{p\in\Sigma_c} s_p(a) + \sum_{p\in\Sigma_u} x_p(a+1) \le c_l.$$
  end for
end for

of a kernel. $R_{\mathrm{FCR}}(b)$ introduces $b|P|$ more artificial variables $z_p(a)$. There exist algorithms introducing fewer variables while still giving a feasible solution to FCR(b). We now introduce the concept of *level* in the following paragraphs and show how it incorporates timing benefits with fewer additional variables.

The concept of level is motivated by the constraint (7). Notice that $u(a)$ is determined by the flow variations on the paths with the longest $w^{\max}_p$. Hence, we can partition P into several level sets and assign each path a level π_p according to the level set it belongs to. Each level π associates with a required time $\overline{w^{\max}_\pi}$ such that $u(a) > \overline{w^{\max}_{\pi-1}}$ while any flow along a level π path changes between step a and $a+1$. In other words, each path p in level $\pi_p = \pi$ satisfies $\overline{w^{\max}_{\pi-1}} < w^{\max}_p \le \overline{w^{\max}_\pi}$. We define the level of the paths with the longest required time to be $\pi^{\max}$ and level 0 has $\overline{w^{\max}_0} = 0$. Without loss of generality, we can set $\overline{w^{\max}_{\pi^{\max}}} = \max_{p\in P} w^{\max}_p$ and by definition

$$0 = \overline{w^{\max}_0} < \overline{w^{\max}_1} < \cdots < \overline{w^{\max}_{\pi^{\max}}} = \max_{p\in P} w^{\max}_p.$$

We create the binary level variable $r_\pi(a)$ to indicate whether the level π is the highest level involving flow change between step a and $a+1$. $r_\pi(a) = 1$ if and only if a flow along a level π path changes and all the flows along higher level paths remain the same between step a and $a+1$. We say a level variable r depicts a FCR solution x if $r_\pi(a) = 1$ implies $x_p(a) = x_p(a+1)$ for every path p with $\pi_p > \pi$. By definition, if the level variable r depicts a feasible solution x, we can construct a feasible solution (x,u,z) by setting $z_p(a) = \left(\sum_{i=\pi_p}^{\pi^{\max}} r_i(a)\right)$ and $u(a) = \sum_{i=0}^{\pi^{\max}} \overline{w^{\max}_i} r_i(a)$ for all $p \in P$ and $0 \le a \le b-1$. Hence, finding a feasible (x,u,z) to FCR is equivalent to finding a feasible x and a level variable r depicting x.

We can find a feasible x and its depicting level variable r by the transformed FCR problem with $\pi^{\max}$ levels in b steps (tFCR$\langle\pi^{\max}\rangle(b)$):

$$\begin{aligned}
\min\ & \sum_{a=0}^{b-1} u(a) \\
\text{s.t. }\ & x \in \mathcal{X} && (10)\\
& u(a) = \sum_{i=1}^{\pi^{\max}} \overline{w^{\max}_i}\cdot r_i(a) && \forall 0 \le a \le b-1 \\
& r_0(a) + \sum_{i=1}^{\pi^{\max}} r_i(a) = 1 && \forall 0 \le a \le b-1 \\
& \left(\sum_{i=\pi_p}^{\pi^{\max}} r_i(a)\right)\cdot \alpha_p \ge |x_p(a+1) - x_p(a)| \\
& && \forall p \in P, 0 \le a \le b-1 \\
& r_i(a) \in \{0,1\} && \forall 0 \le i \le \pi^{\max}, \\
& && 0 \le a \le b-1 \qquad (11)
\end{aligned}$$

where the constraint (10) represents the constraints (3) - (6).

Consider the subproblem tFCRs$\langle\pi^{\max}\rangle(b)$ which assumes $u(a) > 0$ (or $r_0(a) = 0$) for all steps. We form tFCRs$\langle\pi^{\max}\rangle(b)$ by removing all $r_0(a)$ from tFCR$\langle\pi^{\max}\rangle(b)$. Then we propose the **kernel $R_{\mathbf{tFCRs}\langle\boldsymbol{\pi^{\max}}\rangle}(\boldsymbol{b})$**, which solves the linear relaxation of tFCRs$\langle\pi^{\max}\rangle(b)$ (relax the constraint (11)), uprounds the non-zero $r_i(a)$ with the highest level to 1 (the other level variables are set to 0) and sets $u(a)$ to its corresponding $\overline{w^{\max}_i}$ for each a. $R_{\mathrm{tFCRs}}\langle\pi^{\max}\rangle(b)$ solves a feasible r for every feasible x, and hence gives feasible u and z to FCR(b). Theorem 2 suggests that $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle\pi^{\max}\rangle}](b)$ is also a $\frac{\max_{p\in P} w^{\max}_p}{\min_{p\in P} w^{\max}_p}$-approximation algorithm. Nevertheless, we add only $b\pi^{\max}$ more variables in $R_{\mathrm{tFCRs}\langle\pi^{\max}\rangle}(b)$ instead of $b|P|$ in $R_{\mathrm{FCR}}(b)$. In practice, kernel $R_{\mathrm{tFCRs}\langle\pi^{\max}\rangle}(b)$ can even perform near-optimal (much better than $R_{\mathrm{FCR}}(b)$) as shown in Section V-A and V-B.

We know that $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 1\rangle}](b)$ gives the least-step solution. Under order-oblivious scenario, it gives the same update steps as the SWAN solution [15]. SWAN considers no timing information, so its solutions wait at least $\max_{p\in P} w^{\max}_p$ between steps, which equals to $\overline{w^{\max}_1}$. Thus $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 1\rangle}](b)$ performs at least the same as SWAN. Since a multilevel solution can be truncated to a single level solution, we have $\mathrm{SOL}_{ALG[R_{\mathrm{tFCRs}\langle 1\rangle}](b)} \ge \mathrm{SOL}_{ALG[R_{\mathrm{tFCRs}\langle\pi\rangle}](b)}$ if $\pi > 1$. Thus, by considering multilevel $\pi > 1$, $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle\pi\rangle}](b)$ outperforms SWAN.

V. Simulations

We have implemented Algorithm 1 and 2 with two different kernels mentioned in Section IV and the two state-of-the-art methods regarding transient congestion-free reconfiguration on ns-3.24 [25]. Together with the optimal solution, they are summarized below:

- $\mathrm{OPT}_{\mathrm{FCR}}$: The optimal FCR solution obtained by solving the MILP problem.
- $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}]$: Algorithm 1 with the LP-based kernel solving the linear-relaxed transformed FCR subproblem involving 3 levels (set $\pi^{\max} = 3$, delete $r_0(a)$ and relax the constraint (11)).
- $\mathrm{ALG}[R_{\mathrm{FCR}}]$: Algorithm 1 with the LP-based kernel solving the linear-relaxed FCR problem (relax the constraint (9)).
- SWAN: The order-oblivious solution in [15] with the step waiting time equal to the maximum path required time.
- zUpdate: The switch-based routing method proposed in [22] for layered structured networks.

Those algorithms are installed on a controller which communicates with other OpenFlow switches via OpenFlow protocol 0.8.9 [26], which is the latest supported version in ns-3.24, with Multiprotocol Label Switching (MPLS) extension. Controller conducts tunnel-based routing via extended switches supporting Weighted Cost Multipath (WCMP). For the algorithms, CBC-2.9.0 [27] serves as the LP/MILP solver. All measurements are obtained from a 2.4GHz quad core laptop with 8 GB memory on Fedora 20.

We compare the algorithms with the step upper bound $b = 10$. The initial and target configurations for the experiments in Section V-B and V-C result from user and traffic generation. Users are set by a Bernoulli process: each source-destination pair can be selected as a user with a specified probability. For each selected user, 2 acyclic paths are predetermined by Yen's k-shortest-path Algorithm [28] with $k = 2$ to send traffic through. Each user uses UDP to send 1 kb packets at a constant data rate which distributes uniformly between 0 and 1 Mbps. We vary the data rate to form the initial and target configurations.

Each link capacity is set as $1/(1-\lambda)$ times the maximum traffic the link might carry under both configurations. As such, a scratch capacity rate λ for every link is guaranteed, and hence a congestion-free update sequence exists as shown in [15]. The scratch capacity rate results from the fact that the backbone links are in general underutilized [29]. We alter λ in each case to compare the algorithms under different scenarios.

Table III shows some attributes of the algorithms we will compare in this section. We first examine Example 3 in Section V-A to compare the performance of the algorithms. In Section V-B, we show how our algorithm helps a practical large scale inter-data center network achieve faster reconfiguration in reasonable time. Finally, we demonstrate how uncertainty can cause congestion significantly in a data center network with layered structure in Section V-C.

A. A Simple Example

We solve Example 3 by four applicable algorithms (zUpdate is excluded because it requires equal delay between layers), and the results are shown in Table IV.

$\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ attains the 3-step optimal solution as given by $\mathrm{OPT}_{\mathrm{FCR}}(10)$. $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ not only introduces less variables but also achieves a better solution than $\mathrm{ALG}[R_{\mathrm{FCR}}](10)$. By comparing $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ with SWAN, we find that the update time can be shortened with the help of the timing information.

TABLE III
COMPARISON OF THE METHODS

Method	$\mathrm{ALG}[R_{\mathrm{tFCRs}\langle\cdot\rangle}]$	SWAN	zUpdate
Applicable Network	arbitrary	arbitrary	layered structure
Update Objective	minimum time	minimum step	minimum step
Applicable Routing	tunnel-based	tunnel-based	switch-based
Uncertainty Tolerance	yes	yes	no

TABLE IV
PERFORMANCE COMPARISON

Method	Solution Steps	Update Time (unit)
$\mathrm{OPT}_{\mathrm{FCR}}(10)$	10	9
$\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$	3	9
$\mathrm{ALG}[R_{\mathrm{FCR}}](10)$	2	22
SWAN	2	22

(a) The B4 topology

(b) The geographical distribution of the Google Data Centers

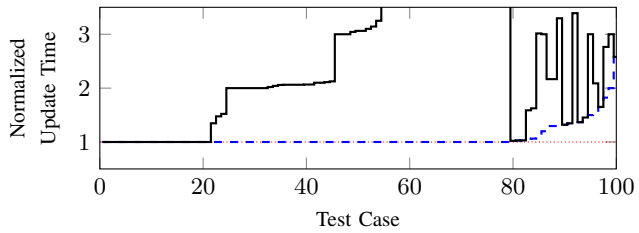
Fig. 4. The B4 topology of 12 data centers

By Theorem 2, the theoretical approximation ratio is $\frac{11}{3}$. In this case, $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ achieves the ratio 1 and $\mathrm{ALG}[R_{\mathrm{FCR}}](10)$ achieves $\frac{22}{9}$.

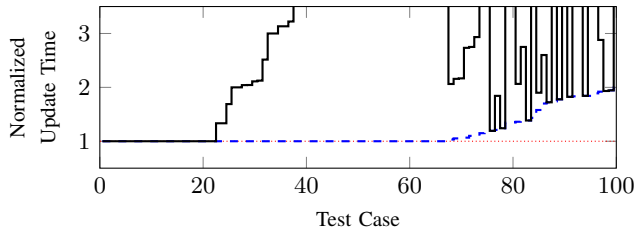
B. WAN (Inter-Datacenter Network)

Our method is also applicable for wide area networks (WANs), such as Google B4 [30]. Google implements B4 to connect their data centers [31]. We create a network consisting of 12 nodes representing those data centers and 19 interconnected links based on the topology described in [30] (Fig. 4(a)) with the link latency (ms) proportional to their actual geographical distance as shown in [31] (Fig. 4(b)). We assume the data centers perform packet switching within a millisecond, which contributes to the uncertainty intervals.

100 random traffic patterns are generated for both $\lambda = 10\%$ and $\lambda = 5\%$ by adding source-destination pair with 0.05 probability. We solve the patterns by $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$, SWAN and $\mathrm{OPT}_{\mathrm{FCR}}(10)$. We know $\mathrm{OPT}_{\mathrm{FCR}}(10)$ gives the shortest update time in 10 steps. Thus for each test case, we normalize the results of $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ and SWAN by $\mathrm{OPT}_{\mathrm{FCR}}(10)$ if they have solutions (Fig. 5). We sort the test cases by the normalized update time of $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$.



(a) Scratch capacity rate $\lambda = 10\%$



(b) Scratch capacity rate $\lambda = 5\%$

Fig. 5. The resulted update time normalized by the shortest update time. Dashed line: $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$; normal line: SWAN; dotted constant 1 line: $\mathrm{OPT}_{\mathrm{FCR}}(10)$

When $\lambda = 10\%$, $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ updates strictly faster than SWAN in 70 out of 100 cases. Also, we can find that $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ takes less than 2 times the shortest update time given by $\mathrm{OPT}_{\mathrm{FCR}}(10)$ in general (Fig. 5(a)).

Whilst $\lambda = 5\%$, there exists no guarantee that we can find a congestion-free update plan in 10 steps. We collect 100 solvable cases and 11 unsolvable ones. SWAN fails in all 11 unsolvable cases, while our method $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ and the optimal method $\mathrm{OPT}_{\mathrm{FCR}}(10)$ can still provide congestion-free reconfiguration update plans for 7 cases because of the timing information. In 69% of the solvable cases, $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ strictly outperforms SWAN (Fig. 5(b)). Again, we can observe that the normalized update time of $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$ is mostly less than 2, even though Theorem 2 promises $\frac{W^{\max}}{W^{\min}}$ only[2].

To verify the congestion-free property, we adjust the buffer size of every network card interface to be only two packets in ns-3 (one in progress and one arriving). For each solution, we monitor the packet drop event. No packet is dropped during the reconfiguration, which implies those methods are truly congestion-free.

This simple example shows how timing information enables us to expand the feasible solution set and update faster. In fact, SWAN considers only the order-oblivious case when the network is totally uncertain for the operator, which is just an extreme case of our framework.

C. Layered Structure (Intra-Datacenter Network)

For a layered network, zUpdate [22] searches for a switch-based least step congestion-free update sequence toward a target set of configurations described by constraints, which can be the target state. The flows from the same user do not interfere with each other, resulting from the assumptions of the layered structure and the absence of uncertainty. Hence, it is another special case of our framework with uncertainty issue eliminated. In practice, rule change will not take effect immediately and thus the deviation can cause congestion during the transient stage.

Fat-tree topology [32] is a layered network structure proposed for data center networks. We implement a simple fat-tree network with 1 ms delay links, as shown in Fig. 6(a), to verify the uncertainty effect. Each circle mark represents a switch and the users are located at the squares. We select source-destination pairs as users with probability 0.1. The scratch capacity rate is set to $\lambda = 17\%$ and we find congestion-free update plans in 5 steps.

We consider two timing uncertainty effects: rule-update processing delay and packet switching delay. When we update the rules of an user at a switch, we encounter a processing delay uniformly distributed over $[0, \delta_d]$ (ms); as a flow arrives at a switch $v \in V$, it gets delayed by a time uniformly distributed over $[0, \delta_v]$ (ms) before it leaves the output interface. We apply both $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$ and zUpdate to solve for congestion-free update plans under $\delta_d = 0.5$ and $\delta_v = 10$. Our method is tunnel-based, while zUpdate reconfigures the network switch-by-switch. We assume further that once a new rule-update instruction is set to a switch, the switch discards the previous in-progress update and starts adopting the new rules.

The time domain simulations are done in MATLAB. We do not simulate the link congestion phenomenon, such as buffering or packet dropping, since those decisions are operator-dependent. Instead, we simply allow flows to exceed the link capacity and we define the *utilization* of a link as the total flow on the link divided by its capacity. When the utilization is greater than one, it implies congestion occurs in the network (not necessarily on the corresponding link).

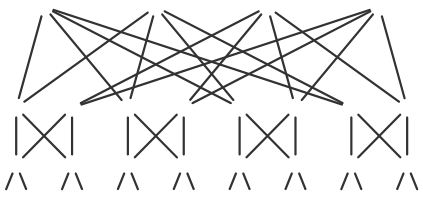
Both $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$ and zUpdate are examined under three different network conditions: without uncertainty, low uncertainty and high uncertainty. We simulate the zUpdate solution without uncertainty, pick the most utilized link during the reconfiguration and show its utilization along the time under each scenario. The simulation results are shown in Fig. 6(b), 6(c) and 6(d). The left charts are the update results of $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$, while the right ones belong to zUpdate.

Both $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$ and zUpdate are congestion-free without uncertainty (Fig. 6(b)). However, uncertainty may result in timing deviation and cause congestion for the zUpdate solution (Fig. 6(c)). The less precise control we can achieve, the more congested situation we will encounter. In all three uncertainty scenarios, we can still update without congestion by applying our algorithm $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$. It ensures congestion-free property during the whole reconfiguration by updating in a slower pace than zUpdate.

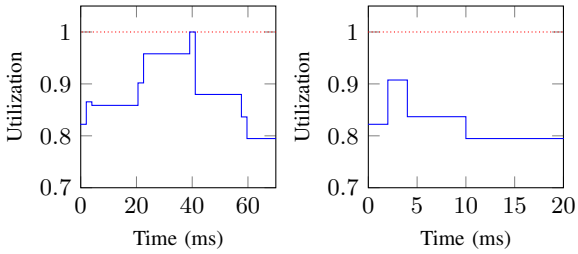
VI. Conclusion

We formulate a time-aware optimization model to find fast congestion-free routing reconfiguration plans. Our approach benefit from given timing information with any level of uncertainty. Several existing models become special cases of our formulation when we have perfect timing information or no timing information at all. This framework helps determine

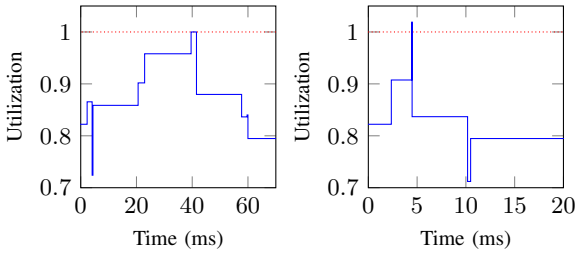
[2]The bound is actually tight for $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](10)$. The algorithm tends to choose smaller latency/capacity ratio instead of shorter latency, and hence we can construct an example showing that the bound is tight.



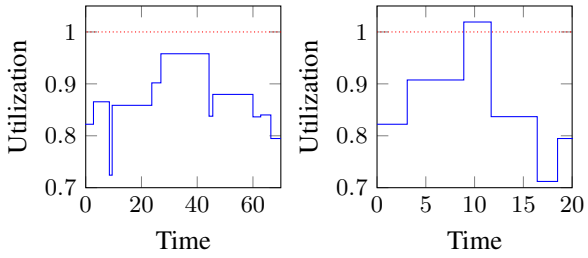
(a) The fat-tree topology



(b) Without uncertainty ($\delta_d = 0, \delta_v = 0$): Both $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$ (left chart) and zUpdate (right chart) are congestion-free



(c) Low uncertainty ($\delta_d = 0.5, \delta_v = 0.5$): $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$ (left chart) is congestion-free while zUpdate (left chart) congests



(d) High uncertainty ($\delta_d = 10, \delta_v = 0.5$): $\mathrm{ALG}[R_{\mathrm{tFCRs}\langle 3\rangle}](5)$ (left chart) remains congestion-free and zUpdate (right chart) endures a long congestion period

Fig. 6. The network topology and the timing charts of the busiest link utilization

less conservative update schedule. We further provide an efficient approximation algorithm to solve this new optimization problem, which is proven to be NP-hard, with performance guarantee. Extensive packet-level simulations confirm our predictions.

REFERENCES

[1] J. Sherry *et al.*, "Making middleboxes someone else's problem: Network processing as a cloud service," *ACM SIGCOMM CCR*, vol. 42, no. 4, pp. 13–24, 2012.

[2] Z. A. Qazi *et al.*, "SIMPLE-fying middlebox policy enforcement using SDN," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 27–38, 2013.

[3] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. USENIX NSDI*, 2005, pp. 273–286.

[4] A. Strunk, "Costs of virtual machine live migration: A survey," in *IEEE SERVICES*, 2012, pp. 323–329.

[5] A. Markopoulou *et al.*, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, 2008.

[6] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. 25, no. 1, pp. 73–85, Jan 1977.

[7] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, no. 2, pp. 97–133, 1973.

[8] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 118–124, 2002.

[9] J. Moy, "RFC 2328: OSPF version 2," 1998.

[10] M. Meyer and J. Vasseur, "RFC 5712: MPLS traffic engineering soft preemption," 2010.

[11] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, vol. 2, 2000, pp. 519–528.

[12] A. Pathak *et al.*, "Latency inflation with MPLS-based traffic engineering," in *Proc. ACM IMC*, 2011, pp. 463–472.

[13] M. Reitblatt *et al.*, "Abstractions for network update," in *Proc. ACM SIGCOMM*, 2012, pp. 323–334.

[14] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates," in *Proc. ACM SIGCOMM HotSDN Workshop*, 2013, pp. 49–54.

[15] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 15–26, 2013.

[16] L. Vanbever *et al.*, "Lossless migrations of link-state IGPs," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1842–1855, 2012.

[17] Summary of the Amazon EC2 and Amazon RDS service disruption in the US East region. [Online]. Available: http://aws.amazon.com/message/65648/

[18] M. Alizadeh *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM SIGCOMM*, 2014, pp. 503–514.

[19] N. L. Van Adrichem *et al.*, "OpenNetMon: Network monitoring in OpenFlow software-defined networks," in *IEEE/IFIP NOMS*, 2014.

[20] M. Azizi, R. Benaini, and M. B. Mamoun, "Delay measurement in OpenFlow-enabled MPLS-TP network," *Modern Applied Science*, vol. 9, no. 3, pp. 90–101, 2015.

[21] C. Yu *et al.*, "Software-defined latency monitoring in data center networks," in *Passive and Active Measurement*. Springer, 2015, pp. 360–372.

[22] H. H. Liu *et al.*, "zUpdate: Updating data center networks with zero loss," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 411–422, 2013.

[23] X. Jin *et al.*, "Dynamic scheduling of network updates," in *Proc. ACM SIGCOMM*, 2014, pp. 539–550.

[24] C. L. Lim *et al.*, "Packet clustering introduced by routers: Modeling, analysis and experiments," in *Proc. IEEE CISS*, 2014.

[25] ns-3. [Online]. Available: https://www.nsnam.org/

[26] OpenFlow switch specification 0.8.9. [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v0.8.9.pdf

[27] CBC (COIN-OR branch and cut). [Online]. Available: https://projects.coin-or.org/Cbc

[28] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[29] A. Hassidim *et al.*, "Network utilization: The flow view," in *Proc. IEEE INFOCOM*, 2013, pp. 1429–1437.

[30] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 3–14, 2013.

[31] Google data center locations. [Online]. Available: http://www.google.com/about/datacenters/inside/locations/index.html

[32] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM CCR*, vol. 38, no. 4, pp. 63–74, 2008.

Scalable, Self-Healing, and Self-Optimizing Routing Overlays

Olivier Brun, Hassan Hassan and Josselin Vallet
LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
email: {brun, hhassan, jvallet}@laas.fr

***Abstract*—After Internet routing was shown in a number of classic measurement papers to result in paths that are sub-optimal with respect to a number of metrics, routing overlays were proposed as a method for improving performance, without the need to re-engineer the underlying network. In this paper, we present SMART, a self-healing, self-optimizing and highly scalable routing overlay, which has a number of advantages with respect to existing solutions. First, SMART can run with off-the-shelf applications and does not require any kernel modification. In addition, SMART can be widely deployed over a sizable population of routers, because it can quickly learn and efficiently track the optimal path with a limited monitoring effort. We describe the design objectives, the architecture and the implementation of SMART, as well as the online decision methods used for learning the optimal routes. Experimental results demonstrate significant improvements over native IP routing, both in terms of latency and throughput.**

I. Introduction

Current Internet routing protocols may work reasonably well when only "best effort" delivery is required, but the requirements of modern distributed services are typically far more stringent, demanding greater performance and availability of end-to-end routes than these protocols can natively deliver. These services often require continuous operation over time, always maintaining the response time below an acceptable threshold, and even small degradations in their performance can have a considerable business impact, in terms of slowed-down service adoption, lost revenue or even damage to brand reputation.

A number of classic measurement studies (see, e.g., [1], [2]) have revealed that the performance of flows could be significantly improved by choosing alternate paths to the ones proposed by IP (Internet Protocol) routing protocols. In addition, it was also shown that path outages are routine events in the Internet, and that the inter-domain routing protocol BGP (Border Gateway Protocol) reacts and recovers slowly from link/node failures [3], [4], [5], causing path outages that can last for several tens of minutes [6], [2], [7].

The ideal solution would be a complete rethink of the Internet routing infrastructure, doing away with the existing architecture and redesigning it with the benefit of hind-sight about its deficiencies. Unfortunately, the so-called ossification of the Internet prevents even changes that are unanimously recognized as necessary to take place.

Routing overlays have been proposed as an alternative solution that can potentially provide the desirable flexibility and control over the routing infrastructure, without the need to re-engineer the Internet [8], [9], [10], [11]. A routing overlay is formed of end hosts, which are deployed in different spots over the Internet. These nodes monitor the quality of the IP routes between themselves and use this information to decide whether to route packets directly over the IP route or by way of other overlay nodes. A routing overlay therefore enables controlling the path of data through the network without modifying the underlying IP mechanism for computing routes, but just by adding intermediate routing hops into the path taken by packets. In a routing overlay, the endpoints of the information exchange are unchanged from what they would have been in the absence of the overlay, but the route through the network that the data traverse may be quite different.

There are several advantages to the use of routing overlays. Firstly, they can be used to quickly recover from path outages. Indeed, they can exploit the inherent redundancy of the Internet to find an alternate path when an IP route becomes unavailable, even if Internet routing protocols cannot. In addition, routing overlays can also be used to improve the quality of service of data flows by overriding the routes determined by Internet protocols and routing traffic based on metrics directly related to application performances.

In this paper, we present SMART[1], a self-healing, self-optimizing and highly scalable routing overlay that we developed. SMART is self-healing because it is able to quickly detect and recover from path outages. It is self-optimizing because it can discover the optimal routes within the overlay network for service-specific routing metrics. It is highly scalable because it was designed to learn the optimal routes in large overlays with a minimum monitoring effort. Last but not least, SMART was designed to control the path of data of an application through the overlay without the application even being aware that its data flows are routed over the overlay, so that it can work with off-the-shelf applications.

In the following, we describe the design objectives, the

ISBN 978-3-901882-83-8

[1]Self-MAnaging Routing overlay

architecture and the implementation of SMART. Since one of our essential design goals was to build a routing overlay that can be widely deployed over a sizable population of routers, we elaborate on the methods implemented in SMART for discovering optimal routes in large overlay networks with a minimum probing effort. Finally, we also present experimental results obtained with real-world experiments over the Internet. These results demonstrate that it is possible to significantly improve over native IP routing with a modest monitoring effort. Due to the lack of space, we do not present the methods used for assessing the quality of overlay links, but interested readers may refer to [12].

The rest of this paper is organized as follows. In Section II, we discuss the similarities and differences of our routing overlay with existing solutions. Section III is devoted to the description of the architecture and components of our system. In Section IV, we describe the technical mechanisms used for forwarding a packet from its source to its destination. Section V presents the methods used for discovering optimal routes in the overlay with a minimum monitoring effort, whereas experimental results are presented in Section VI. Finally we conclude in Section VII with a brief summary and a description of future work.

II. Similarities and differences with respect to existing solutions

Researchers have successfully used overlay networks to solve problems in various areas. To name but a few of the applications, overlays have been used for self-organization in peer-to-peer networks [13], [14], [15], to implement application-layer multicast [16], [17], [18], [19], and even to provide countermeasures to DDoS attacks [20], [21]. Overlay network technologies are also used by Akamai Inc. for dynamic content delivery [22], [23], [24], [25], [26]. Comprising more than 61,000 servers located over 1,000 networks in 70 countries, the Akamai platform delivers 15-20% of all Web traffic worldwide.

More recently, several frameworks have been proposed for overlaying virtualized Layer-2 networks over Layer-3 networks, such as Virtual Extensible LAN (VXLAN) [27] and Distributed Overlay Virtual Ethernet (DOVE) [28]. The main difference between SMART and these technologies is that they rely on the routes provided by Internet routing protocols, without seeking to control how data flows are routed between end hosts.

In that respect, our system is much more closer to the solutions developed by the Detour and RON (Resilient Overlay Network) projects, which have clearly demonstrated the benefits of moving some of the control over routing into the hands of end-systems. The Detour framework [29] is an in-kernel packet encapsulation and routing architecture designed to support alternate-hop routing, with an emphasis on high performance packet classification and routing. In contrast, the developers of RON have opted for a tighter integration of the application and the overlay network since RON is a software library that programs link against [30]. This approach permits "pure application" overlays with no kernel modifications, and allows the use of application-defined routing metrics. Although the objectives of SMART are similar to those of Detour and RON, SMART has the advantage that it can work with off-the-shelf applications an on standard operating systems. Another major difference is that Detour and RON do not scale very well: as the number of overlay nodes n increases, their costly $O(n^2)$ probing overhead becomes a limiting factor. In practice, a reasonable RON overlay can support only about 50 routers before the probing overhead becomes overwhelming [30].

The latter design objective is shared with a self-aware routing protocol known as the Cognitive Packet Network (CPN) [31], [32]. CPN provides QoS-driven routing, and performs self-improvement in a distributed manner by learning from the experience of special packets, which gather on-line QoS measurements and discover new routes. The routing decisions are made at each node of the network, and they are based on adaptive learning techniques using random neural networks. The application of CPN techniques to peer-to-peer overlay networks has been considered in [33]. More recently, the use of CPN-inspired learning techniques in SMART was investigated in [34]. In the present paper, we describe in much more details the architecture and implementation of SMART, and investigate the relevance of a different approach for learning the optimal overlay routes. In addition, whereas only results related to the round trip delay were reported in [34], we present here some experimental results on bandwidth optimization.

III. Architecture of the Routing Overlay

The overlay network is formed of software routers scattered over the Internet. In our experiment, these routers were executed in Virtual Machines (VM) running in cloud computing platforms, but they can be ran on physical hosts as well.

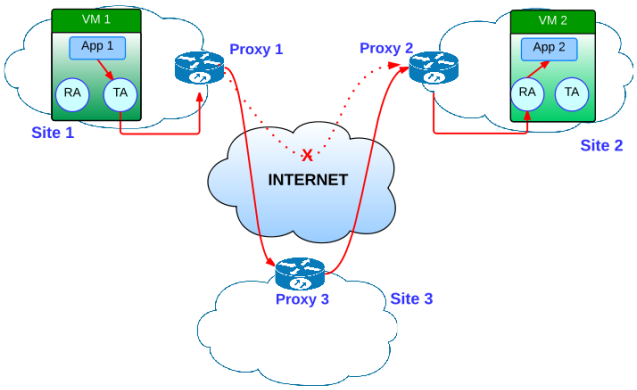


Fig. 1. Architecture of the Autonomic Communication Overlay.

Two types of agents are used. Transmission (TA) and Reception (RA) agents are local agents that are executed on each VM running a task of the distributed application. They represent the entry and exit points of the overlay network, respectively. Each site also runs a single software router called a Proxy. The Proxy is in charge of monitoring the quality of the overlay paths towards certain destinations, selecting the best paths and forwarding the packets of the application over them. As shown in Figure 1, this enables to avoid congested or failed parts of the Internet when a Proxy detects that the IP route is subject to anomalies.

A. Transmission and reception agents

Let us recall that one of our design objectives is to control the path of data of an application through the network, without the application even being aware that its data flows are routed over the overlay. To this end, we use packet interception and encapsulation mechanisms operating in a transparent way for the application. These mechanisms are implemented by two software agents, which are activated automatically at start-up of their respective VM:

- **Transmission agent**: the role of the Transmission Agent (TA) is to intercept the packets sent by the application running in the same VM and to forward them to the local Proxy using IP-in-IP encapsulation.
- **Reception agent**: the role of the Reception Agent (RA) is to receive the packets sent by the local Proxy and to deliver the original packets to the local application running in the same VM.

B. Proxy

An agent, called a Proxy, is executed in each site and acts as an intermediary for communications with other sites. The Proxy is in fact an entity constituted of three different software agents:

- **Monitoring agent**: it monitors the quality of the Internet paths between the local site and the other sites in terms of latency, bandwidth, and loss rate. The monitoring agent can be queried by the routing agent in order to discover the quality of a given path according to a certain metric.
- **Routing agent**: it is configured to optimize a service-specific routing metric towards certain destinations. To this end, it drives the monitoring agent so as to discover an optimal path (e.g., low-latency, high-throughput, etc.) with a minimum monitoring effort (cf. Section V). For each destination, the optimal path towards that destination discovered by the routing agent is written in the routing table of the forwarding agent.
- **Forwarding agent**: it is in charge of forwarding each incoming packet to its destination on the path it was instructed to use by the routing agent.

IV. Packet interception, encapsulation and forwarding

The forwarding of a packet from its source to its destination proceeds as shown in Figure 2.

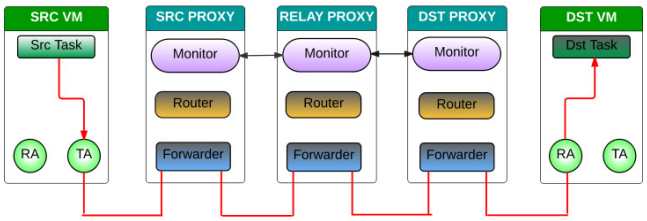


Fig. 2. Forwarding process.

When a packet is sent by a source task to a destination task located in a different site, it is first intercepted and forwarded to the TA. The TA uses IP-in-IP encapsulation to forward an altered packet to the local Proxy. The payload of the altered packet, referred to as the SMART packet in the following, is that of the original packet along with an additional header. Upon reception of the SMART packet, the forwarding agent of the Proxy looks-up its routing table in order to determine the path to the destination. The choice of source routing is dictated by scalability considerations (see Section V). The sequence of intermediate Proxies is written in the SMART header, and then the SMART packet is forwarded to the first one of these Proxies. Each intermediate Proxy then forwards the packet to the next hop on the path, until the final Proxy is reached. When this occurs, the packet is forwarded to the RA of the destination VM. The RA decapsulates the SMART packet and forwards the original IP packet to the destination task using a raw socket. We present below the technical details of each of these operations.

A. Packet interception

The TA intercepts the packets sent by the application running in the same VM and forwards them to the local Proxy. We emphasize that the TA does not intercept all packets, but only packets towards specific destinations located in a different site. The list of destination IP addresses for which packet interception has to be done is controlled dynamically by the routing agent.

As shown in Figure 3, packet interception is realized using a filtering mechanism known as NetFilter NFQUEUE. Netfilter represents a set of hooks inside the Linux kernel [35]. It allows specific kernel modules to register functions that are called back for every packet that traverses the respective hook within the network stack. NFQUEUE is an iptables target, which delegates the decision on packets to user-space software (the TA in our case).

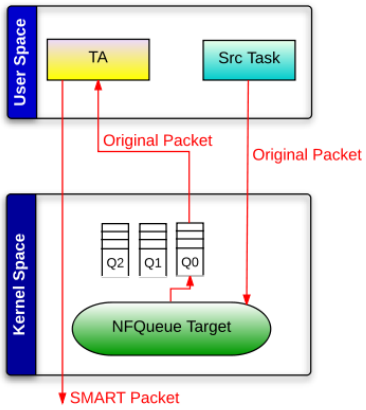


Fig. 3. Forwarding process.

B. Packet encapsulation

Upon receipt of the packet sent by the local application, the TA takes the entire content of the packet received and encapsulates it into its own message format, adding a SMART header that contains control information. It contains in particular the IP address of the destination Proxy (which differs from that of the local Proxy, in the outer IP header) as well as the complete path to reach it, that is, the list of intermediate Proxies. The TA leaves the latter field blank, since the path to the destination Proxy will be determined by the forwarding agent of the source Proxy. Once the header added, the SMART packet is sent to the local proxy using UDP.

C. Processing by the forwarding agent

Upon receipt of a SMART packet, the forwarding agent inspects its header to determine its precise role. There are three cases:

1) The packet is at the source Proxy: this is the case if the Proxy is not the final destination and if the field describing the end-to-end path is blank. In that case, the forwarding agent looks up for the path to the destination Proxy in its routing table, writes this path in the header of the SMART packet, and then forwards it to the next hop on the path.
2) The packet is at an intermediate Proxy: the forwarding agent then just forwards the incoming packet to the next hop on the path, after having updated its destination IP address.
3) The packet has reached the destination Proxy: the forwarding agent then forwards the packet to the RA on the destination VM.

D. Decapsulation and transmission to the destination

The RA decapsulates the Panacea packet and forwards the original data packet to the destination task using a raw socket, that is, an internet socket that allows the direct sending and receiving of IP packets without any protocol-specific transport layer formatting. The packet is directly delivered to the recipient application because the destination IP address is that of the destination VM. We note that there is an additional difficulty when the public IP address of the destination VM is different from its private IP address. In that case, the automatic remapping of public IP addresses into private IP addresses by the Network Address Translation (NAT) mechanism is only possible for the SMART packet, and not for the inner original packet. To overcome this difficulty, the RA uses a configuration file containing translation table entries to convert the public IP address of the packet into a private address. In addition, IP header checksum and any higher-level checksums that include the IP address are also changed by the RA.

E. Packet forwarding overhead

In order to evaluate the time overhead introduced by SMART with respect to native IP routing in a controlled environment, we have used the Common Open Research Emulator (CORE). CORE is an open-source network emulator developed by Boeings Research and Technology division and supported, in part, by the US Naval Research Laboratory [36]. It consists of a GUI for drawing topologies of lightweight virtual machines, emulating end hosts or networking devices (e.g. routers, switches, etc.) running Internet protocols. We have used CORE to emulate linear topologies of different sizes $n = 2, \ldots, 5$ as shown in Figure 4.

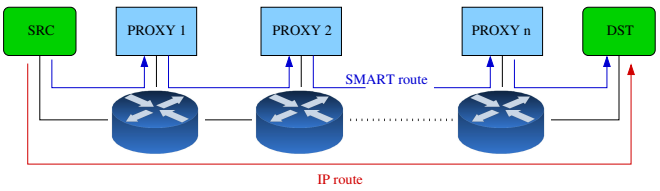


Fig. 4. Experiment to evaluate SMART forwarding overhead.

For each size, we have measured the end-to-end RTT with and without SMART. When SMART is activated, it routes all packets through all available proxies before reaching the destination. We observed an additional end-to-end latency of about 3ms with respect to native IP routing, regardless of the size n of the topology (indicating that most of the overhead is due to the processing done by the TA and RA).

Note also that there is a 28-Byte per-packet overhead for adding the SMART header (20 Bytes) and sending the altered packet with UDP (8 Bytes).

V. Discovering the Optimal Routes

In this section, we assume for simplicity that there is a single origin/destination (OD) pair and describe the algorithm implemented by the source Proxy for learning an optimal route to the destination Proxy. This algorithm is implemented by the Routing Agent of the source Proxy. We assume that at

discrete time steps (say, every minute) the routing algorithm measures the quality of some links and uses this information to decide how to route packets between the source and destination nodes. We define the monitoring effort of the routing algorithm as the number of probed links per time slot. As mentioned in Section II, existing routing overlays use all-pairs probing, which has the advantage that it is guaranteed that an optimal path is discovered; but the downside is that this approach does not scale very well due to its costly $O(n^2)$ monitoring effort in an overlay of n nodes. Since we wish to build a routing overlay that can be widely deployed over a sizable population of routers, instead of requiring an optimal path to be found at each time step, we look for an online decision algorithm that uses a limited monitoring effort but achieves asymptotically the same average (per round) end-to-end performance as the best path. The idea is to design an algorithm that exploits past observations so as to quickly learn and efficiently track the optimal path.

We formulate this problem as a multi-armed bandit problem [37] in which decisions correspond to paths between the source and the destination, and consider it in the adversarial setting where path costs can change arbitrarily from one time step to the other. In this setting, no probabilistic assumption is made regarding the costs of overlay paths, and in particular there is no independence assumption made on these costs. To solve this adversarial bandit problem, we use an algorithm directly inspired from the well-known EXP3 algorithm [38]. At each successive time slot, it chooses a subset of paths to probe, and measures the quality of these paths (e.g., by summing the edge delays if the metric to be optimized is the latency). The algorithm then sends its packet over the minimum-cost path among those it has probed. In other words, probing does not cover *all* possible paths but only a few paths which have been observed in previous probing steps to provide the best performance. However, we have to widen our probing at random over other paths, so that we do not miss out on paths whose quality has substantially improved over recent history. We first give some background information below on the adversarial multi-armed bandit problem, and then present the routing algorithm implemented in SMART.

A. Adversarial Multi-armed Bandit Problem

We represent the overlay network by a complete graph G of n nodes, and we let s and d be the source and destination nodes, respectively. A decision algorithm $\mathcal{A}$ for the multi-armed bandit problem is given as input N paths in G from s to d, indexed from 1 to N. For example, these paths may correspond to the paths of at most two hops between s and d (that is, the direct link and all paths with exactly one intermediate node), in which case $N = n - 1$. The cost of a path i (e.g., its latency, or the inverse of its throughput) may vary arbitrarily over time, but it is assumed to be upper bounded by some constant $\Delta > 0$. At round $t = 1, 2, \ldots$, a cost $\ell_i(t) \in [0, \Delta]$ is assigned to each path i, but it is not revealed to the algorithm. Then, the algorithm chooses a path $i(t) \in \{1, 2, \ldots, N\}$, sends a message over this path and observes its cost $\ell_{i(t)}(t)$. The cumulative cost of the algorithm over T rounds is defined as

$$L_T(\mathcal{A}) = \sum_{t=1}^{T} \ell_{i(t)}(t), \tag{1}$$

whereas the cumulative cost of path i over the T rounds is $L_T(i) = \sum_{t=1}^{T} \ell_i(t)$. The normalized regret of the algorithm $\mathcal{A}$ with respect to the best path is then

$$R_T(\mathcal{A}) = \frac{1}{T}\left(L_T(\mathcal{A}) - \min_{i=1,\ldots,N} L_T(i)\right). \tag{2}$$

The goal is then to design an algorithm $\mathcal{A}$ that perform asymptotically as well as the best path, i.e., such that $R_T(\mathcal{A})$ converges to 0 as T grows to infinity, uniformly over all outcomes sequences.

In [38], Auer et al. gave a randomized algorithm to solve the adversarial multi-armed bandit problem. This algorithm is known as EXP3 and it is based on exponential weighting with a biased estimate of the gains (defined, in our case, as $g_i(t) = \Delta - \ell_i(t)$ for path i), combined with uniform exploration. The regret of this algorithm can be upper-bounded, for any $0 < \delta < 1$, and a fixed time horizon T, with probability at least $1 - \delta$, by

$$R_T(EXP3) \leq \frac{11\Delta}{2}\sqrt{\frac{N\log(N/\delta)}{T}} + \frac{K\log(N)}{2T}. \tag{3}$$

Note that the regret of this algorithm decreases in time according to $1/\sqrt{T}$. We have implemented in the routing agent a slightly modified version of the EXP3 algorithm, which is inspired from the "power of two choices" technique in randomized load-balancing [39]. In this version, precisely described in Algorithm 1, the routing algorithm chooses a subset $I(t) = \{i_1(t), \ldots, i_K(t)\}$ of paths to probe at each round. The path $i_1(t)$ is the IP route from s to d, and the other paths are chosen randomly according to a probability distribution $\mathbf{p}(t)$ that depends on the weights $w_1(t), \ldots, w_N(t)$ of the paths. This distribution is a mixture of the uniform distribution and a distribution which assigns to each path a probability mass exponential in the estimated cumulative gain for that path. Once the paths probed by the Monitoring Agent, the algorithm selects the path $i^*(t)$ with the best performance among those in $I(t)$, and informs the Forwarding Agent that it has to use this path if $i^*(t) \neq i^*(t-1)$. Finally, the algorithm updates the weights of the paths.

It is easy to show that when $K > 1$ this algorithms performs at least as well as EXP3, so that its regret decreases at least as fast as $1/\sqrt{t}$. In practice, with $K = 3$, we often obtain negative values of the regret, indicating that the algorithm performs

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Algorithm 1 Learning optimal paths with the EXP3 algorithm.

Parameters: integer $K \geq 1$; real $\gamma \in (0, 1]$.
Initialization: $w_i(1) = 1$, $i = 1, \ldots, N$.
for $t = 1,\ 2, \ldots$ **do**
  Compute the probability of each path:

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{N} w_j(t)} + \frac{\gamma}{N}$$

  Set $i_1(t)$ to the IP route and choose randomly paths $i_2(t), \ldots, i_K(t)$ according to $\mathbf{p}(t)$.
  Probe the paths $i \in I(t) = \{i_1(t), i_2(t), \ldots, i_K(t)\}$.
  Compute the gains $g_i(t) = \Delta - \ell_i(t)$ for $i \in I(t)$.
  Select the best path $i^*(t) = \arg\max_{i \in I(t)} g_i(t)$.
  Update the weights:

$$w_i(t+1) = \begin{cases} w_i(t)\, exp\left(\gamma \frac{g_i(t)}{N p_i(t)}\right) & i \in I(t), \\ w_i(t) & \text{otherwise.} \end{cases}$$

end for

even better than the best fixed path. If we restrict ourselves to the $N = n-1$ paths of at most two hops, then the monitoring effort of the algorithm is $2K-1$, independently of the size of the overlay network. More generally, with m OD pairs the monitoring effort is $m(2K-1)$, which is less than in the all-pairs probing approach as long as $m < \frac{n(n-1)}{2K-1}$.

VI. Experimental Results

A. Latency minimization

We now describe the results that were obtained with the proposed algorithm during an Internet-scale experiment done in spring 2014, where we used 19 nodes of the $NLNog$ ring[2] shown in Figure 5. Note that these overlay nodes are interconnected by literally hundreds of Internet nodes which are unknown to us or the overlay, and which support the overlay itself.

We first measured the latency between all pairs of nodes every two minutes, communicating through the Internet, for a period of one week using the ICMP-based ping utility. Furthermore, when five consecutive packets were lost between a specific pair of nodes, we considered that the particular source was disconnected from that destination. We thus collected some 1.7×10^6 measurement data over the week, from which we can compute the weighted adjacency matrix of the overlay graph at each measurement epoch, and hence compare the round trip delay of the IP route with that of the optimal overlay route.

The analysis of collected data confirmed the deficiencies of Internet routing observed in previous studies. There was an outage of the IP route at least once in the week for 65% of OD pairs, and 21% of these outages lasted more than 4 minutes (and more than 14 minutes for 11% of them). This analysis also revealed that, as shown in Figure 6, in 50% of the cases it is possible to improve over the latency of the IP route by adding one or more intermediate overlay nodes to the path. Surprisingly enough, in 30% of the cases, the minimum latency path is a path with only one intermediate overlay node, that is, a two-hop path. This shows that a limited deviation from IP actually produces much better QoS than IP itself. Interestingly, even though in 20% of the cases the optimal path is a 3 or 4 overlay-hop path, there is on the average no significant gain (only 5.4%) in considering overlay paths of more than two hops. This suggests that we can restrict ourselves to paths with at most one intermediate overlay node (this is true only on average, since, for instance, the RTT between Narita/Paris can be more than halved if we use two intermediate nodes instead of at most one).

[2]The NLNog ring is a network of 293 nodes scattered over 46 countries (see https://ring.nlnog.net).

Fig. 5. Geographical location of the 20 nodes selected in the NLNog ring.

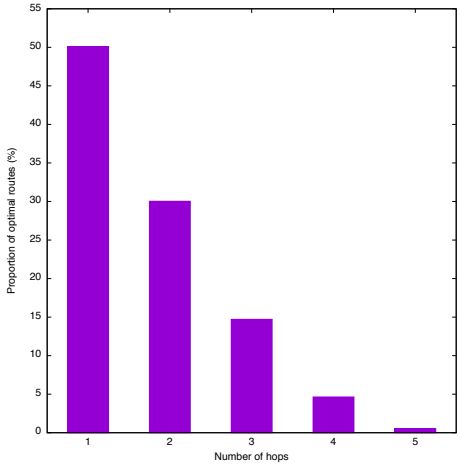


Fig. 6. Percentage of instances when the optimal path includes 1, 2, 3 or 4 hops.

As we will now show, SMART allows a significant decrease in round-trip delay, with a very modest monitoring and computational effort. We consider a fixed OD pair, the other overlay nodes serving just as relays. We restrict ourselves to the $N = 18$ overlay paths of at most two hops and assume that the routing algorithm probes $K = 3$ paths (including the direct IP route) at each time slot, that is, every two minutes.

TABLE I
PERFORMANCE OF NATIVE IP AND SMART ROUTINGS ON THE WHOLE SET OF NLNOG TRACES COMPARED TO OPTIMAL TWO-HOP ROUTING.

	IP route	SMART
Non optimal instants (%)	44.5	3.8
Gap to optimal latency(%)	14.4	0.39

TABLE II
AVERAGE RTT (MS) FOR SOME PATHOLOGICAL OD PAIRS.

	IP route	SMART	OPT 2-hops
Melbourne/Gibraltar	390	274.7	273.5
Narita/Santiago	406.7	254.5	253.0
Moscow/Dublin	179.9	81.9	80.8
Honk Kong/Calgary	267.1	131.8	130.0
Singapore/Paris	322.3	154.9	153.2
Tokyo/Haifa	322.6	180.8	180.1

The algorithm therefore measures 5 links per measurement and decision round (to be compared to the 342 links monitored in the all-pairs probing approach). Our results are summarized in Table I, which shows the average relative gap to the minimum latency that can be achieved with two-hop routing (the averaging is over time and over the 342 OD pairs). Note that this minimum latency corresponds to what would be obtained with a routing overlay using the all-pairs probing approach. These results demonstrate that SMART uses the optimal two-hop route in 96% of the cases, and that it provides near-optimal latencies, with a clear improvement over native IP routing (13.8% on average). However, these average values do not truly measure the gains obtained in the pathological routing situations we seek to improve. In Table II, we present the results for some OD pairs, for which our system allows a huge decrease in round-trip delay.

On the other hand, Figure 7 shows the RTT between Narita (Japan) and Santiago (Chile) over 5 successive days. The RTT of the direct IP route is about 400 ms, whereas the RTT of the minimum latency path is about 250 ms. As can be seen, SMART learns quickly which is the minimum latency path and tracks this path until the end of the 5 days. Figure 8 shows the same results over the first 3 hours. We notice that it takes only 25 measurement epochs (50 minutes) for SMART to learn the optimal route.

B. Throughput maximization

We now describe the results obtained in an experiment involving 9 AWS (Amazon Web Services) data centres located as shown in Figure 9. In summer 2015, we measured the available throughput between all pairs of data centres every five minutes, communicating through the Internet, for a period of four days. We thus collected some 8.3×10^4 measurement data over the 4 days period. Assuming that the available throughput over a path is the minimum of the throughputs of its constituent links, the analysis of these data revealed that the IP route is the maximum throughput route only in 23% of the cases, and that most of the time, the maximum throughput

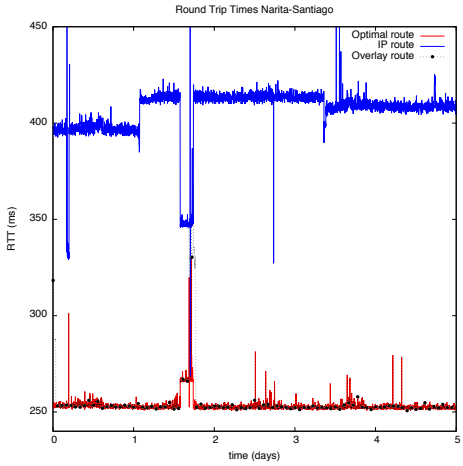


Fig. 7. RTT (ms) measured for the Narita(Japan)-Santiago(Chile) connection in an experiment lasting 5 consecutive days.

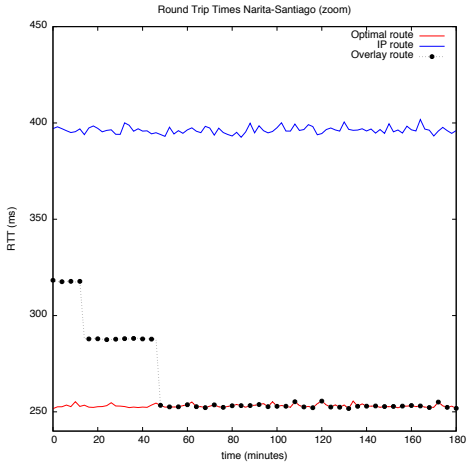


Fig. 8. RTT ms for the Narita(Japan)-Santiago(Chile) connection over the first 3 hours of the experiment reported in Figure 7.

overlay route passes through 1 or 2 intermediate nodes (see Figure 10).

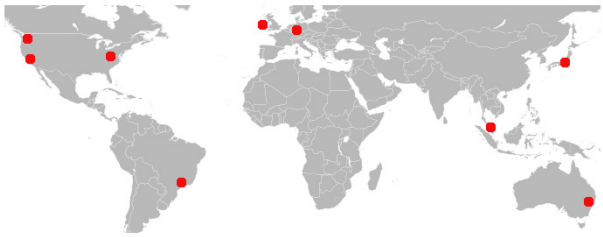


Fig. 9. Geographical location of the 9 AWS data centres.

As in Section VI-A, we consider only the $N = 8$ overlay paths of at most two hops and take $K = 3$. The monitoring effort is therefore limited to 5 links, whereas the all-pair probing measures the throughput of 72 links at each measurement epoch. Our results are summarized in Table III. As

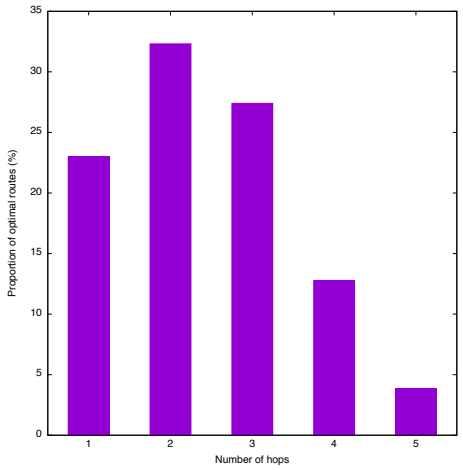


Fig. 10. Percentage of instances when the optimal path includes 1, 2, 3, 4 or 5 hops.

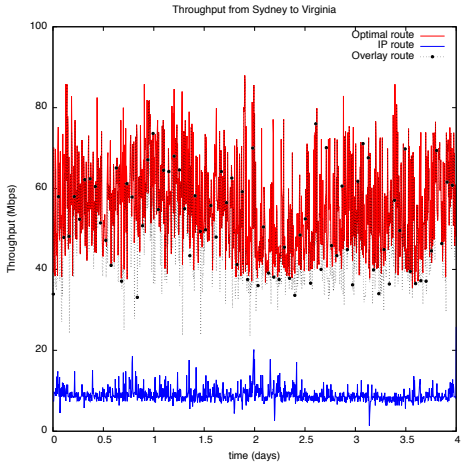


Fig. 11. Throughput (Mbps) measured from Sydney (Australia) to Virginia (USA) over 4 consecutive days.

TABLE III
PERFORMANCE OF NATIVE IP AND SMART ROUTINGS ON THE WHOLE SET OF AWS TRACES COMPARED TO OPTIMAL TWO-HOP ROUTING.

	IP route	SMART
Non optimal instants (%)	73.9	30.1
Gap to optimal latency(%)	31.3	6.6

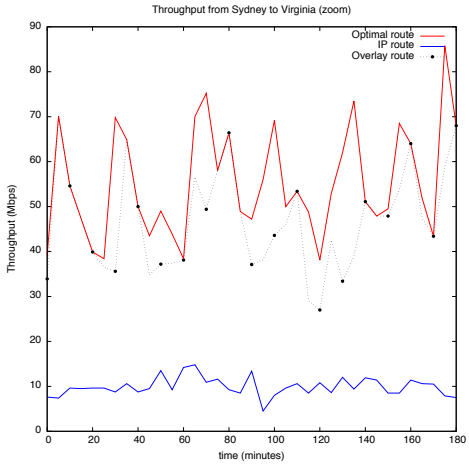


Fig. 12. Throughput (Mbps) measured from Sydney (Australia) to Virginia (USA) over the first 3 hours of the experiment reported in Figure 11.

for the RTT, we observe a clear improvement over native IP routing, and the performance degradation with respect to a routing overlay using the all-pairs probing approach is quite limited (only 6.6%). Here again, we present in Table IV the results obtained for some pathological OD pairs, for which the available throughput is at least doubled.

On the other hand, Figure 11 shows the available throughput between Sydney (Australia) and Virginia (USA) over the 4 successive days. The average throughput of the direct IP route is 8.5 Mbps, whereas the average throughput of the optimal path is 55.3 Mbps. Figure 12 shows the same results over the first 3 hours. We notice that SMART discover an optimal routes almost immediately, but that it is less effective at tracking it than it was the case for the RTT.

VII. CONCLUSION

Internet routing works reasonably well most of the times. Yet, our experimental results show that a routing overlay that make measurement-based online routing decisions can yield spectacular improvements over native IP routing in some cases. The issue is that it is not possible to measure the quality of all overlay links in large overlays, implying that a tradeoff between the quality of the routes discovered and the monitoring effort to discover them is required. To the extend of our knowledge, SMART is the first routing overlay to address this issue.

TABLE IV
AVERAGE THROUGHPUTS (MBPS) FOR SOME PATHOLOGICAL OD PAIRS.

	IP route	SMART	OPT 2-hops
Dublin/Sydney	11.5	35.5	40.5
Singapore/Sao Paulo	12.8	39.5	43.6
Sydney/Virginia	8.5	50.7	55.3
Virginia/Singapore	7.4	31.2	36.1
Virginia/Sydney	6.9	32.2	36.7
Virginia/Tokyo	10.3	37.5	43.4

The results we have obtained have essentially considered paths of at most two overlay hops. Although considerable improvements over native IP routing have been demonstrated, this may not be sufficient for some source to destination pairs. In order to increase the number of potential overlay paths without impairing the convergence time of the learning algorithm, we plan to investigate a different approach based on the so-called *Online Shortest Path Problem* [40]. This approach makes use of the following crucial observation: when the latencies of the edges of some paths are measured, then this also provides some information about the latency of each path sharing common edges with probed paths. As future work,

we intend to study experimentally the performance of this approach, as well as to investigate its generalization to non-additive metrics for the *Online Widest Path Problem*.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PANACEA Project (www.panacea-cloud.eu), grant agreement no 610764.

We wish to thank the administrators of the NLNog ring for providing us access to this platform.

REFERENCES

[1] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of internet path selection," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 289–299, Aug. 1999. [Online]. Available: http://doi.acm.org/10.1145/316194.316233

[2] V. Paxson, "End-to-end routing behavior in the internet," in *in Proc. ACM SIGCOMM'96*, Stanford, CA, USA, August 1996, pp. 25–38.

[3] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed internet routing convergence," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 175–187, Aug. 2000. [Online]. Available: http://doi.acm.org/10.1145/347057.347428

[4] M. Dahlin, B. Chandra, L. Gao, and A. Nayate, "End-to-end wan service availability," in *In Proc. 3rd USITS*, 2001, pp. 97–108.

[5] J. Han and F. Jahanian, "Impact of path diversity on multi-homed and overlay networks," in *In Proceedings of IEEE International Conference on Dependable Systems and Networks*, 2004.

[6] C. Labovitz, R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 515–526, 1998.

[7] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating internet routing instabilities," in *Proceedings of the ACM SIGCOMM 2004 Conference (SIGCOMM), Portland, Oregon, USA*, August 2004.

[8] L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," in *in Proceedings of the 3rd ACM Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[9] J. Touch, Y. Wang, L. Eggert, and G. Finn, "A virtual internet architecture," ISI, Tech. Rep. ISI-TR-2003-570, March 2003.

[10] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, A. Press, Ed., 2004.

[11] M. Beck, T. Moore, and J. Plank, "An end-to-end approach to globally scalable programmable networking," in *in Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, A. Press, Ed., 2003.

[12] U. Ayesta, O. Brun, H. Hassan, and B. Prabhu, "D2.3 - autonomic communication overlay," Deliverable of the FP7 PANACEA project (www.panacea-cloud.eu), Tech. Rep., 2015.

[13] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM'01*, San Diego, California, USA., August 27-31 2001.

[14] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *In the Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, 2001.

[15] B. Y. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, 2003.

[16] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS 2000*, ACM, Ed., Santa Clara, CA, June 2000, pp. 1–12.

[17] S. Banerjee, B. Bhattacharjee, C. Kommareddy, and G. Varghese, "Scalable application layer multicast," in *Proc. of the ACM SIGCOMM*, New York, USA, 2002.

[18] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "Almi: An application level multicast infrastructure," in *Proc of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS)*, San Francisco, CA, USA, March 2001.

[19] J. Liebeherr and T. K. Beam, "Hypercast: A protocol for maintaining multicast group members in a logical hypercube topology," in *Proceedings of the First International COST264 Workshop on Networked Group Communication*. Springer-Verlag, 1999, pp. 72–89.

[20] R. Stone, "Centertrack: An ip overlay network for tracking dos floods," in *in Proc. USENIX Security Symposium '00*, August 2000.

[21] J. Wang, L. Lu, and A. Chien, "Tolerating denial-of-service attacks using overlay networks - impact of overlay network topology," in *in Proc. First ACM Workshop on Survivable and Self-Regenerative Systems*, 2003.

[22] K. Andreev, B. M. Maggs, A. Meyerson, and R. Sitaraman, "Designing overlay multicast networks for streaming," in *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), San Diego, CA, USA*, June 2003.

[23] H. Rahul, M. Kasbekar, R. Sitaraman, and A. Berger, "Towards realizing the performance and availability benefits of a global overlay network," in *Passive and Active Measurement Conference, Adelaide, Australia*, March 2006.

[24] T. Leighton, "Improving performance on the internet," *Communications of the ACM*, vol. 52, no. 2, February 2009.

[25] E. Nygren, R. K. Sitaraman, and J. Sun., "The akamai network: A platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, July 2010.

[26] R. K. Sitaraman, M. Kasbekar, W. Lichtenstein, and M. Jain, *Overlay Networks: An Akamai Perspective*, ser. In Advanced Content Delivery, Streaming, and Cloud Services, E. Pathan, Sitaraman, and Robinson, Eds. John Wiley & Sons, 2014.

[27] J. Moy, "RFC 7348: Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," Tech. Rep., 2014. [Online]. Available: https://tools.ietf.org/html/rfc7348

[28] R. Moats, "Open dove," https://wiki.opendaylight.org/view/Open_DOVE:Main, 2013.

[29] A. Collins, "The detour framework for packet rerouting," Tech. Rep., 1998.

[30] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '01. New York, NY, USA: ACM, 2001, pp. 131–145. [Online]. Available: http://doi.acm.org/10.1145/502034.502048

[31] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "Towards networks with cognitive packets," in *Proc. 8th Int. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems (IEEE MASCOTS), San Francisco, CA, USA*, August 29-September 1 2000, pp. pp 3–12.

[32] E. Gelenbe and Z. Kazhmaganbetova, "Cognitive packet network for bilateral asymmetric connections," *IEEE Trans. Industrial Informatics*, vol. 10, no. 3, pp. 1717–1725, 2014.

[33] M. Gellman, "Qos routing for real-time traffic," Ph.D. dissertation, Imperial College London, 2007.

[34] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *to appear in IEEE Jour. Selected Areas in Communications (special Issue on Emerging Technologies in Communications - Big data)*, 2016.

[35] "Netfilter/iptables," http://www.netfilter.org/, 2014.

[36] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim, "Core: A real-time network emulator," in *In IEEE Military Communications Conference (MILCOM 2008)*, November 2008, pp. 1–7.

[37] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning and Games*. Cambridge University Press, 2006.

[38] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire, "The non-stochastic multi-armed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[39] M. D. Mitzenmacher, "The power of two choices in randomized load balancing," Ph.D. dissertation, University of California at Berkeley, 1991.

[40] A. Gyorgy, T. Linder, G. Lugosi, and G. Ottucsak, "The on-line shortest path problem under partial monitoring," *Journal of Machine Learning Research*, vol. 8, pp. 2369–2403, 2007.

A Hierarchical Control Plane for Software-Defined Networks-based Industrial Control Systems

Béla Genge and Piroska Haller
Department of Informatics
"Petru Maior" University of Tîrgu Mureş
N. Iorga, No. 1, Tîrgu Mureş, Mureş, Romania, 540088
Email: bela.genge@ing.upm.ro, phaller@upm.ro

***Abstract*—Modern Industrial Control Systems (ICS) integrate advanced solutions from the field of traditional IP networks, i.e., Software-Defined Networks (SDN), in order to increase the security and resilience of communication infrastructures. Despite their clear advantages, such solutions also expose ICS to common cyber threats that may have a dramatic impact on the functioning of critical infrastructures, e.g., the power grid. As a response to these issues, this work develops a novel hierarchical SDN control plane for ICS. The approach builds on the features of a novel SDN controller named OptimalFlow that redesigns the network according to the solutions delivered by an integer linear programming (ILP) optimization problem. The developed ILP problem encapsulates a shortest path routing objective and harmonizes ICS flow requirements including quality of service, security of communications, and reliability. OptimalFlow exposes two communication interfaces to enable a hierarchical control plane. Its northbound interface reduces a complete switch infrastructure to an emulated (software) switch, while its southbound interface connects to an OpenFlow controller to enable the monitoring and control of real/emulated switches. Extensive experimental and numerical results demonstrate the effectiveness of the developed scheme.**

***Index Terms*—Industrial Control Systems, Software-Defined Networks, Resilience, Security, Reliability.**

I. Introduction

THE massive proliferation of traditional Information and Communication Technologies (ICT) into the architecture of Industrial Control Systems (ICS) will constitute a turning point in the operation and functioning of modern ICS. This will provide the building blocks for novel infrastructural paradigms, and will facilitate innovative applications such as robust voltage control, renewable energy programs, and electric vehicles. Despite these clear advantages, however, the pervasive integration of commodity off the shelf ICT hardware and software will also expose ICS to new threats [1], [2], [3]. These may have a significant impact on the functioning of critical infrastructures, e.g., the power grid, and may lead to the failure of services, to economic and, possibly, to human losses. As a response, several recently developed techniques address the security of ICS. Furthermore, the NIST Guide to Industrial Control Systems Security [4] recommends integrating different solutions into a defense-in-depth security strategy. On the other hand, ICS require communication resilience solutions that ensure their normal functioning even in the presence of disturbances such as failure and disruptive cyber attacks.

ISBN 978-3-901882-83-8

To alleviate the aforementioned issues we develop a novel hierarchical SDN control plane for ICS. The proposed scheme provides a scalable solution for ICS distributed across large geographical areas. The approach builds on the features of a novel SDN controller named *OptimalFlow*, which monitors a single SDN domain, and redesigns the network according to the solutions delivered by an integer linear programming (ILP) optimization problem. The developed ILP problem encapsulates a shortest path routing objective and harmonizes ICS flow requirements including quality of service, security, and reliability. *OptimalFlow* exposes two communication interfaces to enable a hierarchical control plane. Its northbound interface reduces a complete switch infrastructure to an emulated (software) SDN switch, which exposes the domain's edge ports to the upper tiers and can be monitored and controlled through the *OpenFlow* protocol. We believe that this is a salient feature of the developed scheme, since it facilitates *OptimalFlow*'s adoption in any installation supporting the *OpenFlow* protocol. *OptimalFlow*'s southbound interface connects to an *OpenFlow* controller, to monitor and control a network of emulated or real SDN switches. In order to minimize the impact of network updates on ICS flows we further propose two algorithms. Algorithm 1 dynamically reduces the set of variables in the optimization problem such that only the flows that are affected by a disturbance are optimally redistributed. Algorithm 2 constructs a dependency graph for network updates in order to avoid link congestion. *OptimalFlow* is implemented in the `Python` language and its effectiveness is verified through experiments conducted with Mininet [5] and with the AIMMS optimization software [6].

The rest of this paper is organized as follows. Related Work is briefly discussed in Section II, while the proposed scheme is presented in Section III. Experimental results are detailed in Section IV and the paper concludes in Section V.

II. Related Work

Several recent studies demonstrated the benefits of SDN-enabled communication infrastructures and identified key challenges in adopting this emerging technology. Yonghong Fu *et al.* [7] developed Orion, a hybrid hierarchical control plane for large-scale networks. Orion defines three planes: the

domain physical network, the tier 0 control plane consisting of area controllers, and the tier 1 control plane consisting of a distributed set of domain controllers. While Orion addresses the routing problem in large-scale multi-domain SDN infrastructures, *OptimalFlow* focuses on the requirements of ICS communications including security, reliability and resilience to disturbances. Therefore, similarly to Orion, *OptimalFlow* proposes a hierarchical control plane architecture, but expands the routing criteria from traditional ICT with those specific to ICS. Tuncer *et al.* [8] developed an SDN-based management and control framework for backbone networks. The approach followed a hierarchical and modular structure to support large-scale topologies and the simple integration of various management applications. The work of Tuncer also proposed a network planning algorithm based on the uncapacitated facility location problem. In [9] the authors developed Dionysus, a system for consistent network updates in SDN. Dionysus builds the graph of network update dependencies and schedules these updates by taking into account the performances of network switches. To eliminate packet losses [10] proposed zUpdate, a solution that uses packet labeling for zero packet losses during network updates. In comparison to these works, we believe that *OptimalFlow*, on one hand, and Dionysus and zUpdate on the other hand expose complementary features. Particularly, the hierarchical control plane and the network optimization problem proposed in this work could be extended with Dionysus and zUpdate and their ability to provision network updates with minimum (zero) packet losses.

In the industrial sector, Goodney, *et al.* [11] showed the high degree of network flexibility that can be achieved by adopting SDN for phasor measurement unit (PMU) communications. The benefits of industrial SDN were further demonstrated in a test infrastructure comprising IEC61850-based electrical system [12]. Finally, the work of Dorsch, *et al.* [13] analyzed the advantages and the possible disadvantages of adopting SDN in industrial networks. The authors of [13] acknowledge the benefits of network management applications, quality of service optimization and the enhancement in the system's resilience, but raise serious concerns pertaining to the increased risks of cyber attacks against SDN's centralized controllers.

III. Proposed Approach

A. Architectural Overview

Nowadays, industrial operators are moving towards the adoption of advanced networking solutions from the field of traditional IP networking in order to increase the security and resilience of communication infrastructures. Solutions including Multi Protocol Label Switching (MPLS) [14] and Software-Defined Networks (SDN) [15] have recently been integrated into ICS and have replaced older implementations based on Frame Relay and Asynchronous Transfer Mode (ATM). Nevertheless, communications in large-scale ICS usually cross the boundaries of one administrative domain. In fact, in order to deliver fault-tolerant communications, several lines may be leased from different Internet Service Providers (ISP). Traffic crossing an ISP's networking infrastructure will

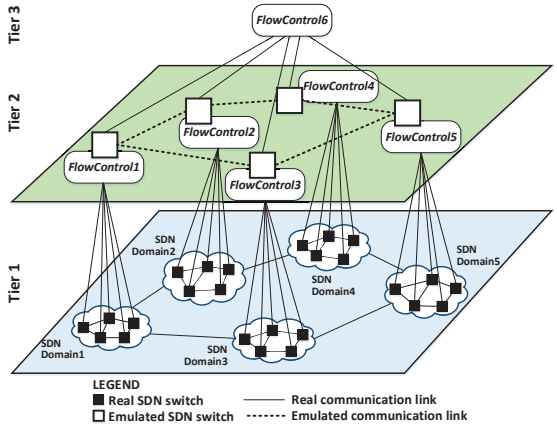


Fig. 1. Architectural overview of the proposed scheme.

therefore be subject to the constraints and routing decision specific to each domain. Based on these assumptions we propose a hierarchical SDN controller scheme that embraces the multi-domain characteristic of ICS networks by means of an n-Tier controller architecture, as shown in Fig. 1. The bottom tier represents the physical infrastructure and consists of network switches and links. This represents the data forwarding plane and can be structured in several domains. Each SDN domain includes a *FlowControl* unit that: (i) monitors the underlying domain for changes in network parameter values, e.g., the status of switch ports; (ii) changes the set of installed flows according to the solutions delivered by an optimization problem aimed to preserve critical communication parameters; and (iii) transparently exposes the edge ports of an entire SDN domain to the upper tiers by means of an emulated SDN switch accessible through the *OpenFlow* protocol.

B. The FlowControl Unit

The *FlowControl* unit (depicted in Fig. 2) includes two software controllers: *OpenFlow* and *OptimalFlow*. The *OpenFlow* controller is a traditional SDN controller that monitors and controls an underlying SDN network using the *OpenFlow* protocol. The *OpenFlow* controller configures the forwarding plane of SDN switches and exposes a communication interface that may be used to implement specially-tailored network traffic control strategies. The main contribution of this work, however, lies in the architecture and in the features exposed by the *OptimalFlow* controller. *OptimalFlow* implements a novel network traffic optimization problem that, as a response to disturbances, computes a new optimal distribution of the affected flows, while preserving the requirements of ICS flows, e.g., security, reliability. Architecturally, it implements four main modules: *SDNStateHandler*, *OFControllerCommunication*, *OptimalSolver*, and *OpenFlowSwitchEmulator*. Its main module is the *SDNStateHandler*, which maintains an in-memory representation of the underlying SDN network and repeatedly issues calls to the *OFControllerCommunication* module to update its internal state. In the case a change is detected, it issues a call to the *OptimalSolver* module

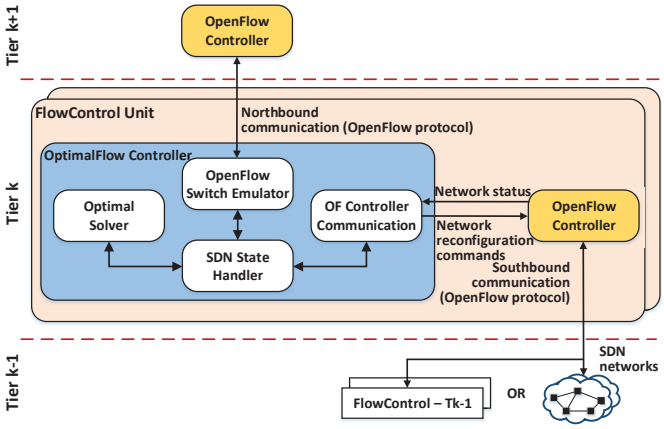


Fig. 2. Detailed architecture of the FlowControl unit.

TABLE I
KEY NOTATIONS

	Symbol	Description
Sets	I, J, B	Flows, SDN switches, and links.
	S	Security features.
Parameters	d_i	Demand of flow i.
	u^b_{jl}	Capacity of link (j,l,b).
	x^A_{ij}, x^E_{ji}	Access and egress flow connectivity.
	p^s_i	Security property requirements for flow i.
	y^{bs}_{jl}	Security properties of link (j,l,b).
	q_i	Minimum reliability requirement for flow i.
	r^b_{jl}	Link failure probability.
	α_i	Penalty value for disconnected flow i.
Variables	t^{bi}_{jl}	Selection of flow i for routing on link (j,l,b).
	w^A_{ij}, w^E_{ji}	Selection of flow i for routing betw. acc./egr. switch j.
	o_i	Selection of flow i for disconnection.

to compute the optimal distribution of the flows affected by the disturbance. The new network configuration is then transmitted by the *OFControllerCommunication* module to the *OpenFlow* controller via a set of static flows that are installed in the SDN switches. The *OptimalFlow* controller exposes an *OpenFlow* northbound communication interface via its *OpenFlowSwitchEmulator* module. By doing so, the *OptimalFlow* controller is connected to upper tiers as a regular SDN switch that can be monitored and controlled via the *OpenFlow* protocol. This represents an effective strategy to build a hierarchical SDN network, where each tier adopts the same *FlowControl* software units. Furthermore, we believe that this is a salient feature of the proposed scheme, since it facilitates the provisioning of *FlowControl* without the need to change the *OpenFlow* protocol and the implementation of SDN switches/controllers.

C. Network Model and Optimization Problem

We assume a demand matrix of flows routed between access and egress switches. The routing needs to be performed in such a way to reduce communication delays by means of selecting the shortest paths and the largest capacity links. Flows are assumed to be non-bifurcated multicommodity flows such that each flow can only be routed on one path. This is a fundamental requirement to ensure that flows may cross the boundaries of one administrative domain. A summary of key notations is tabulated in Table I.

We define I to be the set of flows and J to be the set of switches. Network devices, especially in the industrial sector, are usually connected by more than one link, i.e., by primary and back-up links. Therefore, we define B to be the set of possible links that may connect two switches, and we use (j,l,b) to denote link b between switches j and l, where $b \in B$ and $j,l \in J$. Links between switches may implement various security features such as basic packet filtering (traditional firewalls), encrypted communication links, signature/anomaly detection systems. At the same time, flows may require packets to be forwarded on links with specific security features in place. Therefore, we define S as the set of security features and we use $s \in S$ to denote a specific security feature.

Next, we define the ILP problem's parameters. Let d_i denote the demand of flow i and u^b_{jl} the capacity of link (j,l,b). We assume that if switches j and l are not connected, then $u^b_{jl} = 0, \forall b \in B$. Then, let x^A_{ij} be a binary parameter with value 1 if the access end-point of flow i is connected to switch j, and x^E_{ji} a binary parameter with value 1 if the egress end-point of flow i is connected to switch j.

The security requirements of flow i are configured with the help of the binary parameter p^s_i. This is 1 if flow i may be routed on a link with security property s, and is 0, otherwise. On the other hand, the security properties that are actually installed on a particular link are defined with the binary parameter y^{bs}_{jl}, which is 1 if link (j,l,b) implements the security property s, and is 0, otherwise. In the problem at hand we assume that flow i can be routed on link (j,l,b) only if at least one of the security properties configured for flow i is implemented on link (j,l,b). This means that, for instance, a flow that requires only the integrity security property (s_1), may also be routed on a link that implements other properties as well (s_2, where s_1 is included in s_2), such as integrity, confidentiality. The minimum required reliability of a forwarding path for flow i is defined as parameter q_i, which is a real number bounded between 0 and 1. The probability of link failure is defined as parameter r^b_{jl}, which is a real number bounded between 0 and 1. Finally, we define the α_i parameter as a penalty associated with disconnecting flows. In the problem at hand flows may be disconnected and not routed in the case of significant network failures, which effectively reduce the network's resources and its ability to route flows. The choice of α_i values will lead to a ranking in the significance of flows and it should be chosen significantly larger than the maximum possible value of the objective function's first part, as discussed later in this section.

Next, we define the problem's variables. Let t^{bi}_{jl} be a binary variable with value 1 if flow i is routed on link (j,l,b). Let w^A_{ij} be a binary variable with value 1 if the access end-point of flow i is routed by switch j, and 0 otherwise, and the binary variable w^E_{ji} with value 1 if the egress end-point of flow i

is routed by switch j, and 0 otherwise. We further define the binary variable o_i with value 1 if flow i is not routed due to the unavailability of communication resources, e.g., unavailable bandwidth, and 0, otherwise.

The objective of the optimization is to select the shortest routing path for each flow, while selecting the links with the largest capacities. On one hand, this is accomplished by adopting a *minimization* objective that selects the minimum number of communication links needed to route flows across an SDN domain, i.e., minimize $\sum d_i(t^{bi}_{jl} + t^{bi}_{lj})$. On the other hand, we assume the relative load on a particular link given by the formula $\frac{\sum d_i(t^{bi}_{jl}+t^{bi}_{lj})}{u^b_{jl}}$, which will ensure that the solution includes the links with the maximum capacity. The objective function's second part controls the activation of penalty values in the case of disconnected flows. Therefore, if $o_i = 1$, it activates the penalty value α_i within the objective function. Since α_i is configured as a large integer, the minimization objective will set $o_i = 1$ only if flow i can no longer be routed due to insufficient resources. The objective function is thus defined as follows:

$$\min \sum_{j,l \in J, b \in B} \left(F(j,l,b) \sum_{i \in I} d_i(t^{bi}_{jl} + t^{bi}_{lj}) \right) + \sum_{i \in I} \alpha_i o_i, \quad (1)$$

where $F(j,l,b)$ is 0 if $u^b_{jl} = 0$, and is $\frac{1}{u^b_{jl}}$, otherwise. For the LP at hand the following constraints are defined:

$$w^A_{ij} \leq x^A_{ij}, w^E_{ji} \leq x^E_{ji}, \quad \forall i \in I, j \in J \quad (2)$$

$$\sum_{j \in J} w^A_{ij} \leq 1, \sum_{j \in J} w^E_{ji} \leq 1, \quad \forall i \in I \quad (3)$$

$$\sum_{j \in J} w^A_{ij} = 1 - o_i, \forall i \in I \quad (4)$$

$$w^A_{ij} - w^E_{ji} - \sum_{l \in J, b \in B} \left(t^{bi}_{jl} - t^{bi}_{lj}\right) = 0, \quad \forall j \in J, i \in I \quad (5)$$

$$\sum_{i \in I} d_i(t^{bi}_{jl} + t^{bi}_{lj}) \leq u^b_{jl}, \quad \forall j,l \in J, b \in B \quad (6)$$

$$\sum_{s \in S} p^s_i y^{bs}_{jl} \geq t^{bi}_{jl}, \quad \forall i \in I, j,l \in J, b \in B \quad (7)$$

$$\prod_{j,l \in J, b \in B} (1 - r^b_{jl} t^{bi}_{jl}) \geq q_i, \quad \forall i \in I \quad (8)$$

Constraints (2) enforce that access and egress end-points are only routed by the possible switches, while constraints (3) enforce that each flow end-point is routed by only one switch. Constraints (4) ensure that in the case of insufficient resources flows are disconnected, i.e., $w^A_{ij} = 0$, $o_i = 1$. Constraints (5) denote classical multicommodity flow conservation constraints [16], which impose the selection of a continuous path between access and egress connection endpoints. Constraints (6) impose that the bandwidth required to route flows on link (j,l,b) does not exceed the link capacity. Constraints (7) ensure that flow i is only routed on link (j,l,b) if there exists at least one security property s configured in p^s_i that is implemented on link (j,l,b) and is configured in parameter y^{bs}_{jl}.

Constraints (8) are classical (serial) reliability conditions imposing that the reliability of communication links on a particular path satisfy the minimum reliability requirement q_i. However, we note that the multiplication of several t variables in the reliability constraint (8) will yield a non-linear problem. Therefore, we apply the transformations proposed in [17] to derive a linear set of equations. We observe that since t^{bi}_{jl} is a binary variable, then $(1-r^b_{jl}t^{bi}_{jl})$ can be rewritten as $(1-r^b_{jl})^{t^{bi}_{jl}}$. As a result, constraint (8) is redefined as:

$$\prod_{j,l \in J, b \in B} (1 - r^b_{jl})^{t^{bi}_{jl}} \geq q_i, \quad \forall i \in I \quad (9)$$

By applying the natural logarithm function on both sides of inequality (9) we obtain the following linear reliability constraints, which are adopted in the proposed ILP problem:

$$\sum_{j,l \in J, b \in B} \left[\ln(1 - r^b_{jl}) t^{bi}_{jl}\right] \geq \ln(q_i), \quad \forall i \in I \quad (10)$$

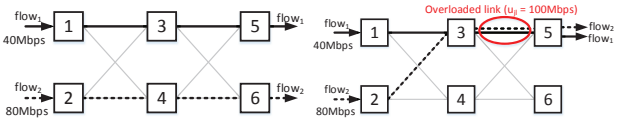
D. Flow Migration Reduction Algorithm

When first launched, *OptimalFlow* computes the optimal mapping of flows on the complete network topology and it configures these flows through an *OpenFlow* controller. However, subsequent runs and most importantly, responses to disturbances might yield a complete network reconfiguration. In fact, such solutions would introduce significant communication disturbances and delays that might disrupt time-critical services and could be easily exploited by attackers. To avoid such scenarios, *OptimalFlow* solves the same optimization problem with a reduced set of variables pertaining to the affected flows. As such, *OptimalFlow* implements Algorithm 1 to reduce the set of variables to those that are affected by disturbances. Initially, the algorithm assumes that all variables are parameters with the value computed in the previous run. Then, for a specific link (j',l',b') on which a disturbance is detected, *OptimalFlow* computes the set of flows AF that are routed on (j',l',b'). For each flow $i \in AF$ *OptimalFlow* adds to the new subset (SV) the variables for the flow's access and egress switches (w^A_{ij}, w^E_{ji}), and for the flow's disconnection (o_i). Then, the algorithm identifies all the communication paths between the flow's access and egress switches. The switches from all the paths SW_i will identify the t^{bi}_{jl} variables, which are added to SV.

While following the above procedure will significantly reduce the impact on communications, after a number of executions the network might require a complete re-organization in order to accommodate all flows and to satisfy the problem's constraints. Therefore, Algorithm 1 is extended with the function CompleteOptimNeeded(sol) that verifies if a complete network reconfiguration is needed. If so, then *OptimalFlow* changes the status of all w^A_{ij}, w^E_{ji}, o_i, and t^{bi}_{jl} parameters to variables and solves the complete network reconfiguration optimization problem. The CompleteOptimNeeded(sol) may be implemented in various ways. For instance, the function might check if $\sum_i o_i > 0$, or if $\sum_i o_i \alpha_i > \beta$, where β is a predefined limit above which a complete network reconfiguration is executed.

Algorithm 1 Flow Migration Reduction

Let (j', l', b') be a failed link.
ChangeOptimizationAllParameters();
AF = GetAffectedFlows(j', l', b');
for each $i \in AF$ **do**
  AddAccessVariable(i);
  AddEgressVariable(i);
  AddDisconnectVariable(i);
  SW_i = GetAccessEgressPaths(i);
  AddSwitchVariables(SW_i);
end for
sol = SolveOptimizationProblem();
if CompleteOptimNeeded(sol) **then**
  ChangeOptimizationAllVariables();
  sol = SolveOptimizationProblem();
end if



(a) Initial topology. (b) Intermediate topology.

Fig. 3. SDN topology reconfiguration with an overloaded link.

Algorithm 2 Dependency Graph Construction

for each $j, l \in J, b \in B$ **do**
  $rescap^b_{jl}$ = GetLinkResidualCapacity(j, l, b);
  $remf^{bi}_{jl}$ = GetLinkRemoveFlows(j, l, b);
  $addf^{bi}_{jl}$ = GetLinkAddFlows(j, l, b);
  if $rescap^b_{jl} \leq \sum_{i \in I} addf^{bi}_{jl} d_i$ **then**
    $dep_{ii'} = 1, \forall i \in I : addf^{bi}_{jl} = 1, \forall i' \in I : remf^{bi'}_{jl} = 1$
  end if
end for
$DEPG$ = InitializeGraph();
OI = GetOrderedFlows();
for each $i \in OI$ **do**
  if IsNotMigrated(i) **then**
    @StepToNextFlow
  end if
  for each $i' \in I$ **do**
    if $dep_{ii'} = 1$ and (i',"REMOVE") $\notin DEPG$ **then**
      AddToGraph($DEPG$,(i',"REMOVE"));
    end if
  end for
  if (i,"REMOVE") $\notin DEPG$ **then**
    AddToGraph($DEPG$,(i,"REMOVE"));
  end if
  AddToGraph($DEPG$,(i,"ADD"));
end for

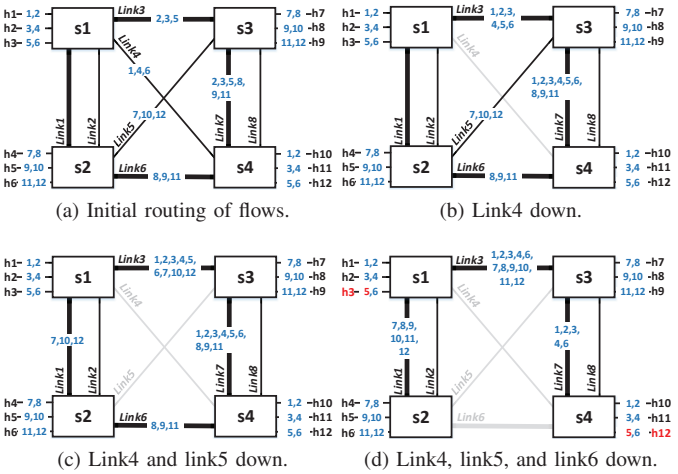
E. Network Update Dependency Graph Construction

The proposed optimization problem generates a new flow configuration that must be provisioned across an SDN installation by the *OptimalFlow* controller. However, the migration of flows from one path to another needs to be carefully planned to ensure that the procedure does not introduce additional disturbances. For example, let us assume the migration of two flows, *flow*$_1$ and *flow*$_2$, with demands of $d_{flow_1} = 40$Mbps and $d_{flow_2} = 80$Mbps, as depicted in Fig. 3. Let us further assume that the capacity of links between switches is of $u^b_{jl} = 100$Mbps, where the number of possible links between switches is equal to one. Initially, the optimization problem routes *flow*$_1$ on links (1,3,1), (3,5,1), and *flow*$_2$ on links (2,4,1), (4,6,1). In the case of a disturbance, however, the migration of any of the two flows before the other is removed may overload links (see Fig. 3 (b)). As a solution, this work adopts a flow migration strategy that relies on the assumption of prioritized flows, which is particularly specific to ICS. For instance, communications in the core of ICS encompassing control hardware connected to critical physical processes may have a higher priority than the communication between human machine interfaces and data servers. The proposed link overload avoidance algorithm, therefore, adopts a priority-based flow migration strategy and builds a network update dependency graph according to the priority of flows. For each flow the algorithm identifies the sequence of flows that need to be deleted before this update can be performed. For instance, by further expanding the previous illustrative scenario, we assume that the priority of *flow*$_1$ is higher than the priority of *flow*$_2$. As a result, before migrating *flow*$_1$, *flow*$_2$ needs to be removed, since *flow*$_1$ has a higher priority. Then, the new paths of *flow*$_1$ and finally of *flow*$_2$'s can be configured in the network switches.

A formal description of the above steps is given in Algorithm 2. At first, the algorithm computes the residual link capacity $rescap^b_{jl}$, it stores in the binary parameter $remf^{bi}_{jl}$ the flows that will be removed from link (j, l, b), and it stores in the binary parameter $addf^{bi}_{jl}$ the new flows that will be routed on link (j, l, b). For each link (j, l, b), if the residual capacity $rescap^b_{jl}$ is lower than the sum of flow demands d_i to be routed on this link, the algorithm adds to the dependency matrix $dep_{ii'}$ of each newly added flow i the flows i' that will be removed from this link. Based on the dependency matrix $dep_{ii'}$, the algorithm then proceeds with the construction of the dependency graph $DEPG$ by first ordering the flows descendingly according to their priorities (set OI). Then, for each $i \in OI$ if i needs to be migrated and a dependency is found with flow i', a `REMOVE` network update is added for flow i' to $DEPG$. Finally, a `REMOVE` update is added for flow i and an `ADD` network update is inserted into $DEPG$.

F. Implementation Details

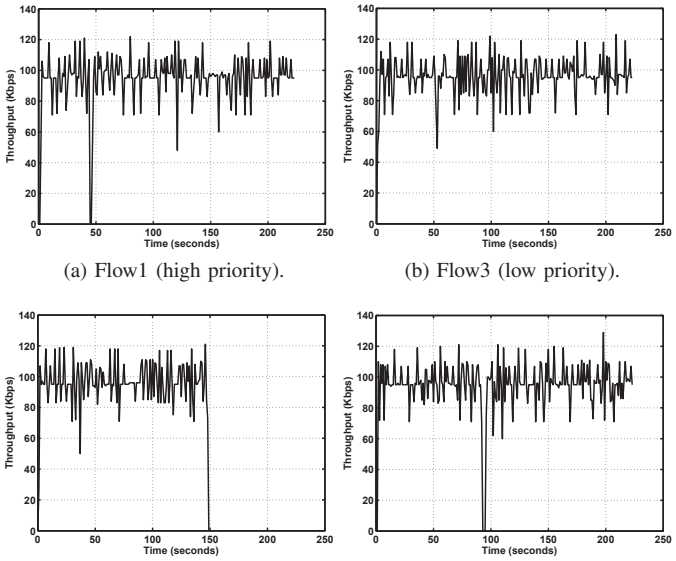
A prototype of the *OptimalFlow* controller was implemented in the `Python` language. *OptimalFlow* integrates part of the POX *OpenFlow* controller's code [18] for emulating an SDN switch. *OptimalFlow* extends three functions in POX's code: `_rx_feature_request()` sends back to the *OpenFlow* controller the status of edge switch ports from an underlying SDN domain; `_flow_mod_add()` calls



(a) Initial routing of flows. (b) Link4 down.

(c) Link4 and link5 down. (d) Link4, link5, and link6 down.

Fig. 4. Network topology and flow routing in the single domain experimental scenario. Switches are connected to the same Floodlight controller and are controlled by one *OptimalFlow* controller (not shown in figure).



(a) Flow1 (high priority). (b) Flow3 (low priority).

(c) Flow5 (low priority). (d) Flow7 (high priority).

Fig. 5. A selection of flows in the single domain experimental scenario. Link4 is disabled at 46s, Link5 is disabled at 92s, and Link6 is disabled at 148s.

OptimalFlow's internal functions to add a new flow; and `_flow_mod_delete()` calls *OptimalFlow*'s internal functions to delete a flow. The *OptimalSolver* module generates an ILP description of the optimization problem and calls the external SCIP (Solving Constraint Integer Programs) solver [19]. Finally, the $OFControllerCommunication$ module uses `pycurl` to communicate with the *Floodlight* [20] controller via its REST API. *OptimalFlow*'s source code is available at http://upm.ro/sereniti/optimalflow.html.

IV. Experimental Results

In order to assess the performance of *OptimalFlow* in various scenarios and problem sizes, we conduct a series of tests including real and simulated settings. First, we perform a qualitative assessment in a scenario with a single SDN domain, which is followed by a scenario with two SDN domains connected in a hierarchical controller scheme. Then, we perform extensive simulations to evaluate the main features and the solutions of the proposed optimization problem. The qualitative tests are performed on an emulated network topology recreated with the Mininet network emulator [5] on Ubuntu LTS 14.04.3 64-bit OS, and a host with Pentium Dual Core 3.00GHz CPU and 4GB of memory. Simulation experiments are performed with the AIMMS software [6].

A. Single Domain Scenario

In the single domain scenario (see Fig. 4) we use Mininet to create a topology of four SDN switches (*s1, s2, s3, s4*) connected to one *Floodlight* controller. The network topology is monitored and controlled by one *OptimalFlow* controller. We assume that switches are inter-connected by two types of links: 600Kbps (denoted by thicker links in Fig. 4) and 300Kbps (denoted by thinner links Fig. 4). We further assume twelve hosts (denoted by *h1, h2, ..., h12*) and a total of twelve `ARP` and `TCP` flows generated with the `iperf` tool. In Fig. 4 flows are numbered from 1 to 12 and are written on the links on which they are routed. Even numbers denote `ARP` flows, while odd numbers denote `TCP` flows. *OptimalFlow* is configured to route twelve flows, where `TCP` flows have a demand of 100Kbps and `ARP` flows have a demand of 1Kbps. For the sake of simplicity we assume the same security and reliability values on all links. We assume that flows 1, 2, 7 and 8 have a higher priority (200.000) than the others (100.000).

Based on the above-defined setting, *OptimalFlow* configures the network with the routing paths depicted in Fig. 4a. Then, at 46s we disable *Link4* (Fig. 4b). *OptimalFlow* solves the network optimization problem by using the variables associated to flows 1, 4 and 6 and by changing the status of the remaining flow's variables to parameters. Since the `TCP` flow 1 is routed on *Link4*, despite its high priority, flow 1 is briefly interrupted (for approx. 1s), an effect, which is shown in Fig. 5a. In the next phase we disable *Link5* at 92s (see Fig. 4c) and then *Link6* at 148s (see Fig. 4d). As shown in Fig. 5 when *Link5* is disabled flow 7 is successfully re-routed. However, by further disabling *Link6 OptimalFlow* concludes that the network does not have the capacity to route all flows. As a result, a low priority flow, i.e., flow 5, is disconnected from the network (Fig. 5c). It should be noted that in this scenario Algorithm 1 reduced the number of re-routed flows since the *Link5* down event caused only flows 7, 10, and 12 to be re-routed, while the path of the remaining flows was not affected. On the other hand, the adoption of priority-based provisioning of flows disconnected a low priority flow, preserving thus the state of critical communication flows.

B. Multi-Domain Scenario

In the multi-domain scenario we create two SDN domains (*DomainA* and *DomainB*) configured identically as the single-domain scenario from the previous section (see Fig. 6). In

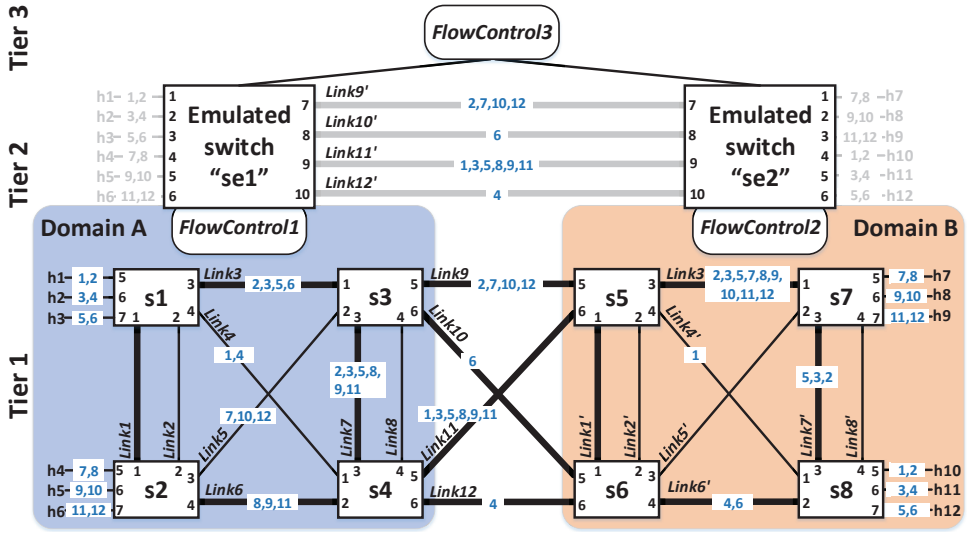
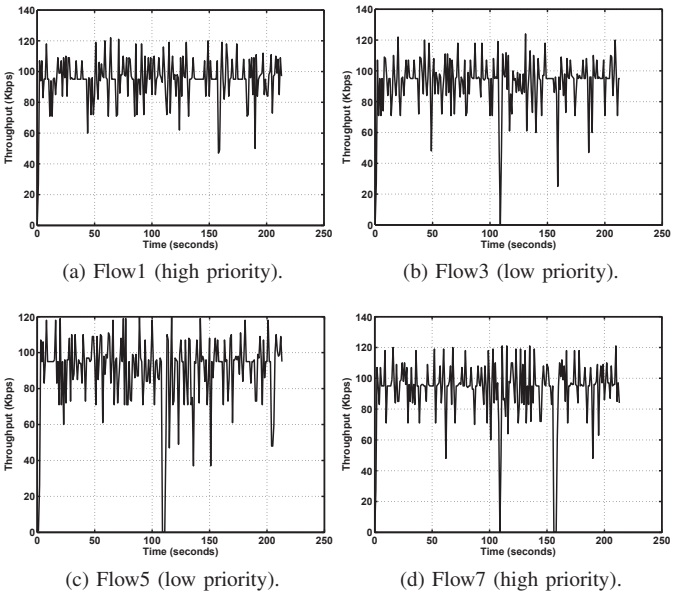


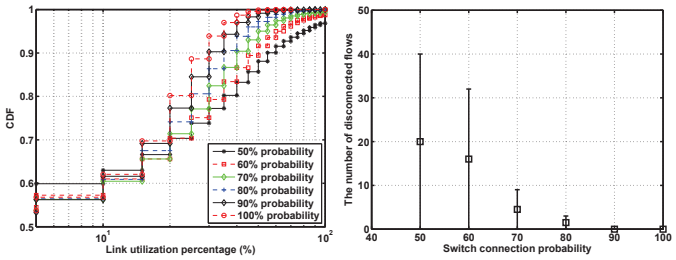
Fig. 6. Network topology and flow routing in the multi-domain experimental scenario.



(a) Flow1 (high priority). (b) Flow3 (low priority).

(c) Flow5 (low priority). (d) Flow7 (high priority).

Fig. 7. Multi-domain experimental scenario: flow throughput. Link3' is disabled at 108s and Link9 is disabled at 155s.

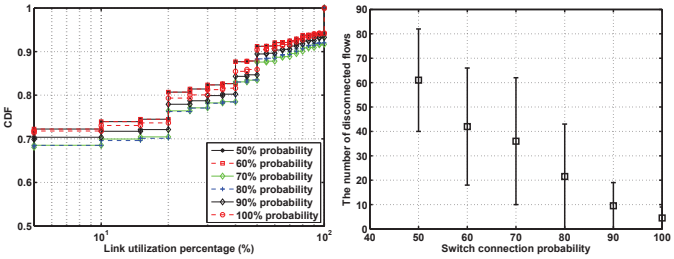


(a) Distribution of link loads. (b) The number of disconnected flows.

Fig. 8. Link load distribution and disconnected flows in the case of equal link capacity (1000Mbps).

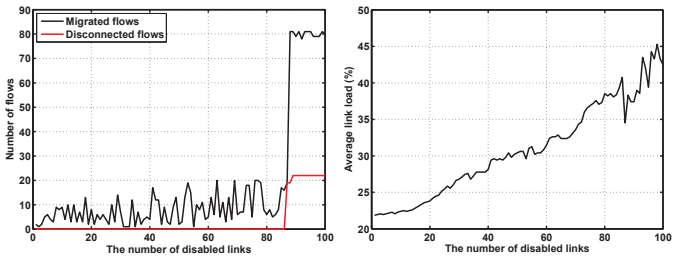
each domain we configure one *Floodlight* controller and one *OptimalFlow* controller. Each *OptimalFlow* controller starts an emulated switch at Tier-2, exposing ten emulated switch (edge) ports to the *FlowControl3* at Tier-3. *FlowControl3* includes one *Floodlight* controller and one *OptimalFlow* controller. The *OptimalFlow* controller's configuration at Tier-3 includes two switches interconnected by four virtual links. The mapping of virtual links at Tier-2 to the real physical links at Tier-1 is described in each *FlowControl*'s configuration file. Each *FlowControl* unit is configured to route the twelve flows as defined in the previous scenario.

First, we start *FlowControl1* and *FlowControl2*. In the initial configuration flows 7, 8, 9, 10 are routed from *DomainA* to *DomainB* via *Link9*, flows 11, 12 via *Link10*, flows 5, 6 via *Link11*, and flows 1, 2, 3, 4 via *Link12*. Next, we start *FlowControl3* to compute a new multi-domain optimal solution. As depicted in Fig. 6, this results in significant changes of routing decisions in the two domains. To start with, flows 2, 7, 12 are migrated to *Link9*, flow 6 is migrated to *Link10*, while flows 1, 3, 8, 9, 11 are migrated to *Link11*. Next, we create two disturbances in the network topology. At 108s we disable *Link3'* in *DomainB*. Since the disturbance does not affect the domain's edge ports, the change is detected by *FlowControl2*, which calculates a new optimal distribution of the affected flows. The disruption is clearly visible in Fig. 7, and more specifically on flow 3 (Fig. 7b), flow 5 (Fig. 7c) and flow 7 (Fig. 7b), which are briefly interrupted. Nevertheless, since flow 1 is routed on *Link4'*, the disturbance does not affect its throughput (see Fig. 7a). Finally, at 155s we disable the inter-domain *Link9*, which briefly interrupts the `TCP` flow 7 and the `ARP` flows 2, 10, and 12. The network change is detected by *FlowControl3*, which solves a new optimization problem and issues inter-domain routing changes. These are received by *FlowControl1* and *FlowControl2*, which also compute a new optimal solution for the affected flows.



(a) Distribution of link loads. (b) The number of disconnected flows.

Fig. 9. Link load distribution and disconnected flows in the case of uniformly distributed link capacity (200Mbps, 500Mbps, and 1000Mbps).



(a) The number of migrated and disconnected flows. (b) The average link load.

Fig. 10. Sequential disconnection of links and their impact on the number of migrated flows, disconnected flows and the average link load.

Nevertheless, despite the two-Tier decision-making process, as depicted in Fig. 7, this does not have a significant impact on flows, which successfully recover after only a brief, e.g., 1-2 second, interruption interval.

C. Quantitative Assessment

By using the AIMMS software we evaluate the solutions of the proposed optimization problem. We assume a large-scale network topology including 50 SDN switches and 100 flows. Switches are structured sequentially in columns of 5; each set of 5 switches is connected to the next set of 5 switches with a certain probability. Flows are routed between the first and the last set of 5 switches. In the first case we assume that $d_i = 50$Mbps, $\forall i \in I$, $|B| = 1$ (one possible link), and $u^b_{jl} = 1000$Mbps. Links and flows are configured with the same security and reliability properties. We test the impact of switch connection probability on the link utilization rate and on the number of disconnected flows. Each configuration is run 50 times and average values are calculated. As shown by results (see Fig. 8a), almost 60% of links are loaded less than 10%. This is a significant aspect in the proposed optimization problem and in the design of a resilient infrastructure where the availability of bandwidth provides the opportunity to migrate flows. Nevertheless, the successful routing of flows also depends on the number of links between switches. As shown in Fig. 8b, with a 50% switch connection probability, on average, we measure 20 disconnected flows (out of 100 flows). However, by increasing the connection probability, at 90% the average disconnected number of flows reduces to 0. Next, we change the scenario and assume a uniform distribution of link capacities of 200Mbps, 500Mbps, and of 1000Mbps. In essence, compared to the first case, we decrease the overall capacity of the communication infrastructure. This effect is visible in Fig. 9a, where in approximately 70% of the cases links are loaded less than 10%. Apparently, in this case only 10% of links exhibit a load higher than 50%, while in the previous case links 15% showed a load higher than 50%. This is explained by the results in Fig. 9b, where the average number of disconnected flows increases to 60 for a 50% switch connection probability, as opposed to the average of 20 in the previous case. Furthermore, despite increasing the connection probability to 100%, there are still flows that cannot be routed due to the decrease in the capacity of paths.

Finally, by using the first configuration as described in this section, we measure the number of migrated flows in the case of randomly disabled links. As shown by the results in Fig. 10a, the number of migrated flows, that is, the number of flows for which a new optimal solution is computed, exhibits a slight increase from 1 to 20 for up to 80 disabled links. This means that *OptimalFlow* only needs to solve the optimization problem for a reduced set of variables, e.g., for the affected flows and for the switches on the paths of access and egress end-points. Nevertheless, after disconnecting 85 links *OptimalFlow* determines that a number of 19 flows cannot be routed, which triggers the execution of the optimization problem on the complete network topology, as described in Algorithm 1. As a result, this increases the number of migrated flows to 80, but keeps the number of disconnected flows at 19, which later increases to 22. By sequentially disabling links we also measure an increase in the average link load. As shown in Fig. 10b the average link load increases up to 40% for 84 disabled links, and decreases down to 35% when the optimization is executed on the complete network topology. Inevitably, by further disabling links, the average link load continues to increase up to 43% for 100 disabled links.

D. Execution Time

The execution time of *OptimalFlow*'s network reconfiguration procedure depends on the size of the network. This translates to the number of variables in the network optimization problem and the number of flows that need to be installed. We generated several network topologies with a different number of flows (N_f), switches (N_s), possible links between switches (N_b), and switch connection probabilities. We measured the time in which *OptimalFlow* solves the complete optimization problem for each of these settings. As denoted by the results in Table II, for $N_f = 20$, $N_s = 30$, $N_b = 1$ and a connection probability of 50% *OptimalFlow* solves the optimization problem in 0.34s. However, by increasing N_b to 3, the solve time also increases to 1.06s. A connection probability of 100% further increases the solve time to 1.28s. At the other end, for $N_f = 100$, $N_s = 50$, $N_b = 3$ and a connection probability of 100% the solver needs 20.20s to generate a solution. These

TABLE II
OPTIMIZATION PROBLEM SOLVE TIME.

		Switch conn. probab=50%		Switch conn. probab=100%	
N_f	N_s	$N_b = 1$	$N_b = 3$	$N_b = 1$	$N_b = 3$
20	30	0.34s	1.06s	0.39s	1.28s
50	30	0.95s	2.85s	1.16s	3.59s
100	30	2.35s	6.53s	2.72s	7.48s
20	50	0.94s	2.92s	1.1s	3.47s
50	50	2.67s	7.79s	3.00s	9.69s
100	50	6.33s	17.25s	7.15s	20.20s

TABLE III
FLOW INSTALLATION TIME.

5 flows	20 flows	40 flows	80 flows	160 flows	240 flows
16.9ms	52.9ms	129.6ms	176.3ms	372.9ms	528.5ms

high execution times, however, are addressed by *OptimalFlow* in several ways: (i) the full network optimization is only solved at network start-up; (ii) a change in the network topology will reduce the set of variables used in the optimization problem to the affected flows and switches; and (iii) *OptimalFlow*'s hierarchical structure specifically targets large-scale infrastructures, in which case network planning techniques [21], [22] may be used to optimize the placement of *OptimalFlow* controllers.

Finally, we measured the time in which *OptimalFlow* builds the dependency graph and pushes static flows to the *Floodlight* controller. We assumed the single-domain scenario, as presented in the previous sections, and that each flow is configured on four switches. As denoted by the results in Table III for 20 flows *OptimalFlow* runs the flow installation procedure in 52.9ms, which increases to 176.3ms for 80 flows and up to 528.5ms for 240 flows. It should be noted, however, that the execution time is linear and can be further decreased by a careful network planning strategy, as described earlier.

V. CONCLUSIONS

We developed a novel hierarchical SDN control plane for ICS. The approach builds on the features of an SDN controller named *OptimalFlow* that addresses various requirements of a modern ICS communication infrastructure including scalability, dynamic network redesign as a response to failure or cyber attacks, harmonized routing decisions that encapsulate quality of service, security and reliability properties of communications. The effectiveness of the developed scheme was tested in various experimental and simulation-based scenarios. As shown by results, *OptimalFlow* can be adopted in single and in multi-domain SDN scenarios. To ensure a high performance, however, the parameters of *OptimalFlow* need to be integrated into network planning solutions, which would yield an optimal distribution of *OptimalFlow* controllers.

ACKNOWLEDGMENT

This research was supported by a Marie Curie FP7 Integration Grant within the 7th European Union Framework Programme (Grant no. PCIG14-GA-2013-631128).

REFERENCES

[1] M. Hagerott, "Stuxnet and the vital role of critical infrastructure operators and engineers," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 4, pp. 244 – 246, 2014.
[2] B. Genge, F. Graur, and P. Haller, "Experimental assessment of network design approaches for protecting Industrial Control Systems," *International Journal of Critical Infrastructure Protection*, vol. 11, pp. 24–38, 2015.
[3] B. Genge and C. Enachescu, "Shovat: Shodan-based vulnerability assessment tool for Internet-facing services," *Security Comm. Networks*, 2015.
[4] V. M. K.Stouffer, S.Lightman and A.Hahn, "Guide to industrial control systems (ics) security," *NIST Special Publication 800-82, Revision2, National Institute of Standards and Technology*, 2015.
[5] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, (New York, NY, USA), pp. 253–264, ACM, 2012.
[6] AIMMS, "Advanced Interactive Multidimensional Modeling System." http://www.aimms.com/aimms/, 2015. [accessed December 2015].
[7] Y. Fu, J. Bi, Z. Chen, K. Gao, B. Zhang, G. Chen, and J. Wu, "A hybrid hierarchical control plane for flow-based large-scale software-defined networks," *Network and Service Management, IEEE Transactions on*, vol. 12, pp. 117–131, June 2015.
[8] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive resource management and control in software defined networks," *Network and Service Management, IEEE Transactions on*, vol. 12, pp. 18–33, March 2015.
[9] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, "Dynamic scheduling of network updates," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 539–550, Aug. 2014.
[10] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz, "zupdate: Updating data center networks with zero loss," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 411–422, Aug. 2013.
[11] A. Goodney, S. Kumar, A. Ravi, and Y. Cho, "Efficient pmu networking with software defined networks," in *Smart Grid Communications, 2013 IEEE International Conference on*, pp. 378–383, Oct 2013.
[12] E. Molina, E. Jacob, J. Matias, N. Moreira, and A. Astarloa, "Using software defined networking to manage and control IEC 61850-based systems," *Comput. Electr. Eng.*, vol. 43, pp. 142 – 154, 2015.
[13] N. Dorsch, F. Kurtz, H. Georg, C. Hagerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *Smart Grid Communications, 2014 IEEE International Conference on*, pp. 422–427, Nov 2014.
[14] IBM and Cisco, "Cisco and IBM provide high-voltage grid operator with increased reliability and manageability of its telecommunication infrastructure," *IBM Case Studies*, 2007.
[15] West Nippon Expressway Company Limited, "Software-defined networking (SDN) solution." http://www.nec.com/en/case/w-nexco/pdf/brochure.pdf, 2015. [Online; accessed December 2015].
[16] C. Meixner, F. Dikbiyik, M. Tornatore, C. Chuah, and B. Mukherjee, "Disaster-resilient virtual-network mapping and adaptation in optical networks," in *Optical Network Design and Modeling (ONDM), 2013 17th International Conference on*, pp. 107–112, April 2013.
[17] C. Chiang, M. Hwang, and Y. Liu, "An alternative formulation for certain fuzzy set-covering problems," *Mathematical and Computer Modelling*, vol. 42, no. 34, pp. 363 – 365, 2005.
[18] "POX openflow controller." http://www.noxrepo.org/pox/about-pox/, 2015. [Online; accessed December 2015].
[19] T. Achterberg, "Scip: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.
[20] "Project Floodlight." http://www.projectfloodlight.org/floodlight/. [Online; accessed December 2015].
[21] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 7–12, ACM, 2012.
[22] B. Genge, P. Haller, and I. Kiss, "Cyber-security-aware network design of industrial control systems," *Systems Journal, IEEE*, vol. PP, no. 99, pp. 1–12, 2015.

DISTTM: Collaborative Traffic Matrix Estimation in Distributed SDN Control Planes

Rhaban Hark*, Dominik Stingl*, Nils Richerzhagen*, Klara Nahrstedt‡ and Ralf Steinmetz*
*Multimedia Communications Engineering Lab, Technische Universität Darmstadt, Germany
‡Department of Computer Science, University of Illinois – Urbana, USA
*{rhaban.hark|dominik.stingl|nils.richterzhagen|ralf.steinmetz}@kom.tu-darmstadt.de ‡klara@illinois.edu

***Abstract*—Recently, several works propose monitoring approaches for the emerged paradigm of Software-defined Networking. These provide a couple of ideas to retrieve various information about the network state leveraging new concepts for monitoring data collection at flow-level. As existing approaches reduce their scope to networks with a single controller, even sophisticated approaches ignore a potentially great efficiency gap, due to redundant flow measurements by multiple controllers in adjacent networks. To show a possibility how to close this efficiency gap, we propose a solution for collaborative traffic matrix estimation, termed DISTTM. It exploits the property that flows traverse multiple networks and are monitored by several controllers. Through collaboration, the resulting monitoring tasks are coordinated and distributed among participating controllers to capture relevant information about all traversing flows, omitting redundant data collection. Conducted simulations reveal that DISTTM operates efficiently: the monitoring traffic is significantly reduced, while the traffic matrix entry staleness is slightly affected. Furthermore, DISTTM provides different schemes for a fair load balancing on controllers and switches while taking different influencing aspects into consideration.**

I. INTRODUCTION

Traffic monitoring constitutes the basis for nearly all network management functions. Since monitoring should always be non-invasive, thus, rather passive, yet robust, accurate and timely, a trade-off between performance and monitoring costs is omnipresent. This makes monitoring an inevitable challenging, however, required task. With the utilization of Software-defined Networking (SDN) [14], that provides a separation of the control and data plane, new possibilities evolve to measure traffic in the data plane. Using new techniques, the need for additional intelligence at forwarding elements and additional dedicated collection devices, as was required in traditional monitoring solutions, such as sFlow[1] or NetFlow [4], vanishes. Lately, several approaches propose a variety of monitoring tasks, leveraging these new techniques of SDN, in particular flow-level counters. These approaches measure, for instance, basic metrics, like link utilizations, delay, and packet loss [3], [21], [22]. Sophisticated approaches collect aggregated information (e.g. traffic matrices [19]) or execute more advanced monitoring tasks, such as heavy hitter detection [7].

So far, current approaches limit the control and measurement of networks to single controllers. However, to (i) avoid a single point of failure, (ii) provide scalability and (iii) relieve a controller from frequent polling of the switches, relying on single controllers is not recommended. As a consequence, physically distributed control planes are taken into account.

This work proposes a first step towards controller collaboration for monitoring in SDNs. The benefits of the controller collaboration are shown at the example of distributed traffic matrix estimation. Traffic matrices provide useful means for network provisioning, route planning, and further management tasks [20]. The proposed approach, called DISTTM, makes use of the property that traffic flows most likely take paths through multiple adjacent network portions. The system hinders the networks to monitor the flows each. More precisely, it avoids the redundant measurement of flows traversing through multiple networks that are managed by different controllers. Instead, using DISTTM, the controllers collaborate to coordinate monitoring tasks and share information in the control plane.

DISTTM can be utilized inside single-administrated domains, such as data center networks (*intra-domain* collaboration), as well as, competing, adjacent domains (*inter-domain* collaboration) and is applicable in both scenarios. As discussed later, controllers must expose only a minimal amount of information to other controllers. In addition, domains are not forced to provide capacity which is not equally provided in return.

One further contribution of this work is a fair distribution of load among controllers and switches. Since the system determines traffic matrices collaboratively, we propose different load distribution schemes to assign responsibilities for flows among participating controllers or switches in a fair manner. As different scenarios require different schemes for load distribution, we show three elementary fairness schemes for different purposes and scenarios.

The conducted simulations of DISTTM show, that the system significantly reduces the overall monitoring traffic overhead. Thus, it lowers the load on the controllers, whereas it affects the accuracy in terms of staleness of matrix entries only slightly. Furthermore, simulations show that we attain a fair distribution of the overhead for switches and controllers.

The remainder of the paper is structured as follows: Section II discusses the relevant background and related approaches. Section III describes the design of DISTTM. Further on, Section IV presents the preliminary evaluation while Section V concludes the paper.

[1] sFlow Overview, http://www.sflow.org/about/ [Access: Oct 21, 2015]

ISBN 978-3-901882-83-8

II. Background and Related Work

This section gives an overview over helpful backgrounds and relevant works which contribute to this paper. The first subsection briefly introduces mechanisms to capture data plane information in SDNs, before the second subsection specifies traffic matrices of interest. Additionally, it gives a short introduction to OpenTM, a traffic matrix estimation approach for single-controller SDNs. Its basic concept of how a traffic matrix can be determined is used as reference for non-collaborating networks. Subsequently, the third subsection describes selected works about collaboration between network nodes. To the best of the authors' knowledge, collaborative monitoring for SDNs on controller level, as it is targeted in this work, has not yet been investigated.

A. Data Capturing in Software-defined Networks

In OpenFlow [12], the widely accepted de-facto standard for Ethernet-based SDNs, the data plane consists of dump switches with flow tables managed by controllers in the control plane. In addition to a number of management fields, these flow tables comprise counter fields which are incremented every time a packet is processed using the corresponding entry. For every entry a packet as well as a byte counter are available for monitoring use. These can be deactivated individually to reduce the load on a switch. The controller is supposed to fetch counters in multiple ways: (i) using explicit statistic requests for single flow entries or aggregations; or (ii) implicitly when a flow entry is removed (e.g. triggered by a timeout). The OpenFlow switch specification version 1.5[2] introduces additional thresholds for counters, providing push-based counter access. A related OpenFlow extension that is denoted FleXam [16] provides additional packet sampling capabilities. In contrast, approaches such as OpenSketch [23] and DCM [24] implement a specialized data plane which make it again necessary to customize the protocol, yet yielding to efficient data collection mechanisms.

Anyway, most existing monitoring approaches, rely on version OpenFlow protocol version 1.3 which is expected to be a stable basis[3]. In order to reach applicability in non-custom SDN environments, this work only uses features available in the original protocol.

B. Traffic Matrices of interest

Traffic matrices are abstract data structures showing traffic information for pairs of network nodes. As an example and also used in this work, a traffic matrix can accumulate the amount of traffic in terms of packets or bytes between all ingress/egress switch pairs in the network. They are used for management tasks like capacity planning, network provisioning, load balancing policies for route optimization, but can also be used to detect traffic anomalies and other security related events [20]. They might contain different representations, such as a maximum, minimum, average or a sum. Besides traffic matrices, other types of matrices, like delay matrices, presenting the node-to-node delay for all pairs, or loss matrices exist. Further studies as well as a taxonomy can be found in [13], [20].

Tootoonchian et al. propose OpenTM [19] to estimate traffic matrices in the context of SDN. It takes advantage of the logical centralization of the controller. More precisely, it uses centrally available information given by the controllers routing application to observe upcoming flows. Further on, it queries a flows path in order to be able to select one switch on the path and poll it for statistics. By accumulating flow level byte counters of flows originating at the same node and ending at the same node, it calculates all entries of the traffic matrix. Aside traffic matrix estimation, OpenTMs main contribution is an intelligent selection of polled switches on the path in order to support a fair overhead distribution among switches and yet perform well in terms of accuracy due to packet loss. However, this work adopts the matrix estimation concept of OpenTM based on available routing information.

C. Collaboration in SDN

In the context of network collaboration, Yu et al. [24] propose a memory efficient collaboration-enabled control plane for SDNs. The work states that flows are often monitored redundantly at different switches if flow aggregation is used to reduce the number of rules. On the other side, if single flows are selected to reach fine granular measurements, the number of rules becomes too large. They tackle the problem using two-stage Bloom filters on switches. These filters can be defined in a way the switches monitor particular sets of flows without the need to define one rule per flow. Thus, as rules can be defined efficiently in alignment with monitoring rules of other switches, DCM allows collaboration on switch level. In contrast, we try to achieve collaboration on the controller level. Terzis et al. [17] already proposed a model for collaboration on a comparable level in 1999 in another context. In their approach, bandwidth brokers of different domains maintain agreements to cooperatively allocate resources for *inter-domain* traffic.

In order to be able to collaborate with other controllers, recent approaches introduce infrastructures for distributed SDN control planes [2], [5], [9], [15], [18]. Their target is to give controllers of the same control plane a shared view on the whole network while distributed properties are abstracted as good as possible for their applications. Those approaches do not include monitoring as a potential controller application for distributed control planes. However, DISCO [15] introduces monitoring agents in controllers which are able to measure link utilization. Actually, as the monitoring is limited to links between peering points to adjacent networks and each controller measures the statistics individually, no collaboration is done in this context. In this work it is assumed that the control plane allows controllers to communicate with one another. Hence, a distributed control plane infrastructure, such as DISCO,

[2]ONF: SDN Resources – Technical Library, https://www.opennetworking.org/sdn-resources/technical-library [Access: Oct 22,2015]

[3]Sean Michael Kerner: OpenFlow Protocol 1.3.0 Approved, http://www.enterprisenetworkingplanet.com/nethub/openflow-protocol-1.3.0-approved.html [Access: Oct 26, 2015]

would be a good basis to satisfy the need for a communication possibility between controllers. Levin et al. [11] point out, that trade-offs between staleness and optimality as well as between application complexity and robustness turn up when logically centralized control planes are mapped to physically decentralized.

III. DISTTM

This section deals with the concept of the distributed collaborative traffic matrix estimation system, termed DISTTM. DISTTM consists of collaborating modules that are installed at multiple controllers and exchange messages among each other for the estimation of traffic matrices at the controllers. Therefore, a controller periodically interacts with its switches to capture the data, as detailed in Section III-A. Given this interaction pattern, the collaboration and coordination among controllers for the distributed estimation of traffic matrices are presented in Section III-B and Section III-C, respectively. Finally, Section III-D introduces three schemes to influence DISTTM's coordination for a fair task distribution based on different criteria.

A. Generation of Traffic Matrices

The problem of estimating a traffic matrix in an SDN with a single controller can be tackled using, for instance, OPENTM [19] as described in Section II-B. A controller informs its traffic matrix estimation module whenever a new flow is installed. Afterwards, the statistics for this flow are periodically polled from a selected switch on the flow's path. The interval for the periodic polling is specified by the system parameter T (*polling request interval*). To select a switch along a flow's path, OPENTM presents multiple strategies for an intelligent and sophisticated switch selection. However, DISTTM relies on a random selection of a switch along the path for the sake of simplicity and directs to OPENTM for questions relating the switch selection. The traffic matrix module in a controller uses the gathered statistics from the selected switch to generate the traffic matrix and update the affected cell of the matrix. For the identification of the correct cell, the controller uses the origin (ingress switch) and the destination (egress switch) of the flow. Given the example in Figure 1, C_A polls statistics of flows f_A, f_B and f_C. Regarding f_A, the statistics can be polled at switch S_{A1}, S_{A3} or S_{A2} along its path. The measurements are used to fill the cell of pair $(S_{A1}, S_{A2}) = (ingress, egress)$. C_B polls statistics of f_B and f_C to fill the cell of pair (S_{B1}, S_{B2}) as well as f_D for (S_{B3}, S_{B2}). C_C acts analogously in its network. A flow is polled until the controller receives a message about its timeout (*FlowRemoved*). In this work, DISTTM stores the total amount of bytes for each cell. Other metrics, such as the throughput or bandwidth consumption, may be calculated or derived as well.

The described interaction between a controller and its switches represents an easy-to-operate solution. However, the simplicity of this approach comes at the expense of inevitable excessive cost in terms of statistic requests at switches and controllers, particularly questioning its scalability. To reduce the overhead and improve scalability, DISTTM breaks up with the necessity to measure each upcoming flow on each controller. Instead, we introduce the concept of controller collaboration, i.e. sharing of information, in the following.

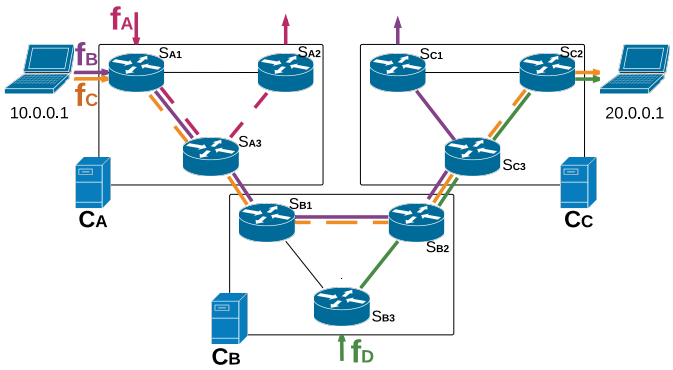


Fig. 1. Inter-network flows.

B. DISTTM *Controller Collaboration*

Assuming the knowledge about peering points between adjacent networks, flows which traverse through networks of cooperating controllers can be identified. As an example, C_A detects, that f_B and f_C exit its network towards the network of C_B. Given this capability and knowledge about traversing flows, DISTTM provides the functionality to coordinate the monitoring. Hence, controllers are able to trigger a coordination whenever considered necessary to distribute responsibilities for redundantly measured flows like f_B and f_C among the controllers. A coordination assigns each flow to exactly one controller, which is subsequently exclusively responsible to capture statistics of this flow (e.g. only C_A captures f_B and C_B captures f_C). So, DISTTM informs each controller (i) about the flows it must monitor and (ii) about other controllers that are interested in the collected statistics of the corresponding flows. As a result of that coordination, in addition to periodic polling of statistics, controllers must transmit updates to a list of interested controllers. These actions are periodically triggered depending on the system parameter T.

Given the example in Figure 1, flow f_B originates at host `10.0.0.1` and has its destination at host `20.0.0.1`, traversing the network of controller C_A, C_B, and C_C. In the naive solution controller C_A, C_B, and C_C would poll statistics of flow f_B from selected switches in their network to update their traffic matrix with the resulting traffic of f_B. Using DISTTM, only one controller, for instance C_A, would fetch the statistics of f_B and provide the information to C_B and C_C. In turn, other flows, such as f_C and f_D, may be assigned to C_B and C_C, while the resulting measurements are transmitted to all interested controllers. As flow f_D only traverses the network of C_B and C_C, C_A is even not aware of f_D and will not be informed about updates of f_D. Flow f_A is only monitored by C_A as it only traverses the network of C_A.

For this particular example the number of statistic requests reduces from 9 ($C_A : [f_A, f_B, f_C] + C_B : [f_B, f_C, f_D] + C_C : [f_B, f_C, f_D] = 9$) to 4 ($C_A : [f_A, f_B] + C_B : [f_C] + C_C : [f_D] = 4$).

A controller uses directly polled values and received values from other controllers to update its traffic matrix. In the example C_A uses the polled values of f_A to update the matrix entry for pair (S_{A1}, S_{A2}). To update the (S_{A1}, S_{A3}) entry it uses the directly polled values of f_B and complements it with the received values from C_B about f_C.

Among the collaborating controllers DISTTM identifies redundantly monitored flows and controllers interested in these flows. Based on these insights DISTTM selects one controller out of the pool of interested controllers to assign the flow to. The chosen controller is subsequently responsible to poll statistic values about the flow and share them with all interested controllers. Hence, only a fraction of all traversing flows must be monitored per controller. Besides the reduction of monitored flows, the controller collaboration also influences the number of transmitted messages as well as of transmitted bytes. Exchanged information about multiple flows between two controllers can be batched into one so-called *statistic batch message*, whereas direct polling of values from multiple flows must be performed per flow. Furthermore, a controller proactively transmits information to interested controllers at the end of each polling interval. This proactive transmission avoids unnecessary transmissions to request the statistics.

So far, DISTTM basically focuses on a distributed and fair procedure for collaborative data collection. As a result, it currently relies on static per-flow polling intervals as well as a static information exchange intervals between controllers. For future work, improvements concerning the polling of values by a controller in its own network, such as flow aggregation [25] or adaptive polling [3], are applicable to minimize the traffic or improve the performance.

C. DISTTM *Coordination*

As described, the system shares responsibilities for flows among collaborating controllers. For this task DISTTM delegates the coordination to one controller which is denoted as *coordinator*. In this work, a static configuration is used to select the coordinator. Nevertheless, leader election algorithms from other research areas (e.g. [1]) are applicable that deal with this problem and allow an autonomous and distributed election of a coordinator. Using a central coordinator simplifies the distribution of flow responsibilities among the controllers, as described hereafter.

1) Coordination trigger: The advantage of DISTTM comes at the expense of coordination overhead. This overhead composes of coordination requests, responsibility assignments and statistic exchange messages. Although the evaluation will show that the impact of additional messages is reasonable, it still needs to be considered. In order to avoid too many coordinations and limit the resulting coordination overhead in networks and scenarios with strong flow fluctuations, DISTTM introduces a counter in every participating controller

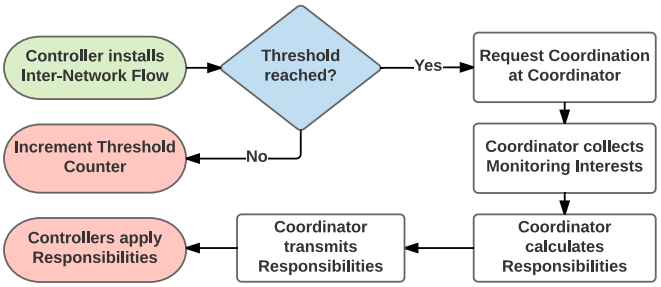


Fig. 2. Coordination mechanism flow diagram.

and a system parameter, referred to as *coordination threshold* R. The threshold may be set dynamically by the controllers based on other system parameters according to findings in the evaluation. At each controller, this counter is incremented every time a controller becomes aware of a new flow. For this, a controller takes only flows into account which have their destination in it's network but do not originate inside of the network. Based on this scheme, simultaneous coordination requests from multiple controllers are avoided since only the last network considers an upcoming flow. In the example given in Figure 1, assuming f_B, f_C and f_D leave the union of cooperating networks at the network controlled by C_C, the threshold counter of C_C would be set to three. If the incremented counter reaches the specified coordination threshold R a coordination procedure is triggered. The coordination procedure comprises the transmission of a so-called *coordination request message* from the controller to the coordinator. The counters are reset every time a coordination procedure is performed.

2) Coordination procedure: As described above, the coordinator executes a coordination procedure whenever it receives a coordination requests. As depicted in Figure 2, if the coordinator receives a coordination request message it collects the monitoring interests of all participating controllers. Therefore, it sends a *monitoring interest request message* to all participating controllers, asking for flows which occurred since the last coordination and traverse their network. The requested information is sent back to the coordinator, using a *monitoring interest response message*. Subsequently, the coordinator applies a *responsibility calculation function* based on the received interests. This responsibility calculation function is responsible to assign flows to the controllers. Furthermore, a list of interested controllers per flow is appended to inform the responsible controller about other interested controllers. The flow assignments including the list of interested controllers are transmitted to the corresponding controllers, using a *coordination instruction message*.

Concerning the example of Figure 1, the coordinator takes the information that controller C_A is interested in f_A, f_B and f_C; C_B and C_C are interested in f_B, f_C and f_D as depicted as input in Figure 3. The function may distribute the responsibilities, for instance, as follows: C_A is responsible to monitor its *intra-network* flow f_A. Furthermore, C_A is responsible to measure f_B and inform C_B and C_C about its

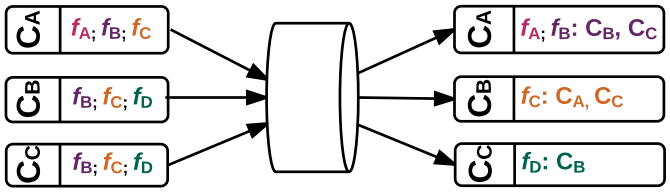


Fig. 3. Coordination function input and output.

updates. C_B needs to monitor f_C and inform C_A and C_C, while C_C is responsible for flow f_D and to inform C_B about measurement updates. In this case, excluding *intra-network* flows, the coordinator assigns one flow to each controller. Hence, it shares the responsibilities fair among controllers.

D. Fairness schemes

In addition to the relevant flows of a controller, a monitoring interest response message comprises further information. This covers the number of *intra-network* flows inside of a single network (e.g., f_A in C_A), the number of previously assigned, still active flows and the number of switches in the controllers network. The additional information is optional and may not be included due to privacy constraints. However, if it is included, the coordinator uses this information for the responsibility calculation function in order to apply various fairness schemes that influence the assignment of flows. DISTTM offers four fairness strategies that are applicable to distribute the responsibilities among controllers.

*1) Fair Controller Distribution **(FCD)**:* The *FCD* scheme distributes the responsibilities equally among all participating controllers in terms of monitored flows in total. Hence, this scheme makes sure that every controller is supposed to make the same number of statistic requests to switches in its network. This includes requests for *intra-network* flows not traversing networks controlled by other controllers. However, *FCD* leads to equal statistic request load among controllers and might be preferred, for instance, by single-administrated networks.

*2) Fair Domain Distribution **(FDD)**:* As different controllers may be part of competing domains, they are not willed to respect *intra-domain* flows of other networks. As a simplified scheme to support this, the *FDD* scheme distributes the responsibilities equally among controllers while *intra-network* flows are not taken into account. This scheme leads to an equal distribution of flows to monitor for other controllers. A more elaborated version of this scheme could consider fairness between each controller pair. A controller C_A, for instance, does neither want to respect a flow only in the network of C_B, nor traversing only the networks of C_B and C_C.

*3) Fair Switch Distribution **(FSD)**:* So far, the issue that networks may differ in size has been ignored. Consider two collaborating networks: one consisting of only two switches, while the other consists of an order of magnitude more switches. If flows traverse both networks with the same probability, a distribution based on the statistic requests a controller has to poll from a switch would lead to a disproportional load on the two switches of the small network. To tackle this issue, the *FSD* scheme assigns the responsibilities in a fair manner based on the current requests per switch. This leads to a fair load distribution among switches in the network, however, potentially not among controllers.

*4) Random **(RD)**:* As reference model, a random distribution among controllers is used. As it can be expected, for large experiments, the scheme resembles the fair controller distribution assuming equal flow occurrences per network.

The listed schemes optimize particular scenarios and situations. Hence, we propose to use dynamic combinations based on the characteristics and requirements in real-world scenarios. As a flow is assigned to a controller DISTTM does not dictate which switch needs to be selected on a flow's path to fetch statistics from. Consequently, the controllers are responsible to make a sensible selection. As already mentioned in Section III-A, related approaches tackle this aspect and is not taken into consideration in this work. However, it may be an important aspect for future work.

IV. EVALUATION

This section describes the evaluation of DISTTM. The major objectives of the evaluation are to show (i) that the system is able to reduce monitoring costs, while the performance can be maintained and (ii) to highlight the influence of the different fairness schemes of the responsibility calculation function on DISTTM. To be able to quantify this, we implemented DISTTM as a Floodlight[4] OpenFlow controller application as which it is deployed on all controllers in a virtual simulation network. In addition, we included a simplified routing application based on a modest host detection extension, the ability to detect peering points with other networks, and a straightforward westbound interface. The westbound interface enables the communication between controllers through TCP connections using an out-of-band controller network in the control plane. Using this small set of functionality, the distributed traffic matrix estimation system – DISTTM – is implemented.

A. Evaluation Methodology and Scenario

We use Mininet [10] as evaluation testbed to generate virtual OpenFlow-enabled networks. As sketched in Figure 4, for this preliminary evaluation, we choose a synthetic, linear topology with three individually controlled networks with three switches each as default scenario. The first network, controlled by C_A has a peering-point to the network controlled by C_B, which itself has a peering point at the other end to the network controlled by C_C. This topology allows a direct presentation and measurement of the DISTTM principles.

For the evaluation, we use the parameters shown in Table I. The first to two parameters represent DISTTM's system parameters, as introduced in the previous section. The following parameters except the last one configure the modeled flows in the network and are detailed in the following paragraph.

[4]Project Floodlight: Floodlight OpenFlow Controller http://www.projectfloodlight.org/floodlight/, [Access: Nov 09, 2015]

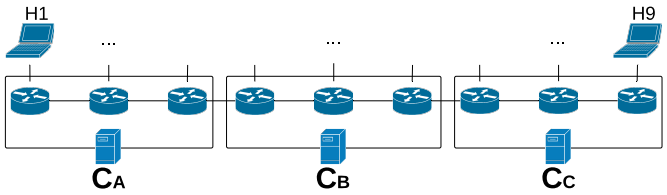


Fig. 4. Synthetic evaluation topology.

The last parameter F_{max} represents the flow idle timeout. If multiple values are listed for a parameter, the values for a default configuration are underlined.

TABLE I
SCENARIO AND SIMULATION SETUP.

Parameter	Values	Description
T [ms]	500, 1000, 2000, 5000	Polling period
R	1, 2, 3, 5, 10, 20	Coordination threshold
r [pps]	100	Packet rate
λ [s]	1	E[Flow inter-arrival time]
k	5	Pareto shape index
x_{min} [pkts]	500	Min. packets per flow
F_{max} [s]	5	Flow idle timeout

As only the duration of active flows instead of their size is relevant for the conducted evaluation, flows have a constant packet rate r and minimal packet size. For the inter arrival time of flows in seconds, we use an exponentially distributed stochastic variable with $\lambda = 1$ [8]. Concerning the length of a flow, a Pareto distributed stochastic variable is used. As parameters, $k = 5$ and $x_{min} = 500$ are set. Thus, most flows are small (slightly larger than 500 packets) and only a few are significantly large. A flow consisting of more than 500 packets is alive for at least 5 seconds with the specified packet rate. Assuming one second inter arrival time, if the system is in equilibrium, at least (and most commonly) six flows intersect at a time. We set the probability of a flow's source (H_i) and destination (H_j) being in one of the networks to $\frac{1}{3}$ each, with $i \neq j$. All simulations were repeated 50 times. Bar plots report the mean with 95% confidence intervals. Box plots report the median, lower and upper quartiles as well as whiskers for the fifth and 95th percentiles. The simulations were executed on a dedicated simulation machine[5].

During the evaluation, we investigated three different metrics to quantify DISTTM's behavior. At first, we measure monitoring costs of DISTTM using the number of statistic requests per controller per flow. This metric is independent of the evaluation duration as the number of flows increases linear with time if the system is in equilibrium. The second metric quantifies the overhead which is additionally produced by DISTTM. This overhead comprises the different types of coordination messages as well as statistic batch messages, which are exchanged between controllers. Similar to the first metric, the second one is measured per controller per flow. To enable a reasonable aggregation of the two metrics, in some cases the metric's presentation is in bytes instead of the number of messages or requests. Although one cannot compare statistic requests and coordination messages directly due to the fact that they do not share the same medium and lead to different load on the controllers, this compromise allows finding a comprehensible operating point for the trade-off between cost reduction and introduced overhead. Note, that the statistic requests with their replies are always of the same size, so that the qualitative behavior is similar for message counts as well as bytes. The resulting bytes of coordination messages may vary depending on the type and amount of shared information. To compare fairness, we selected the portion of statistic requests a controller or switch is supposed to trigger or answer, respectively, as metric. This is identical to the portion of flows a controller must monitor. In addition to these metrics that particularly quantify the costs of DISTTM, it must be shown, that the performance for the generation of the traffic matrices does not suffer using DISTTM. As packet losses are not considered in the scope of this evaluation, the performance is measured in terms of staleness of matrix entries. The staleness is defined by the period between two consecutive updates of a matrix entry for active flows.

[5] *Ubuntu 14.04.3 LTS / 24x2.6GHz Intel(R) Xeon(R) CPU / 128GB RAM*

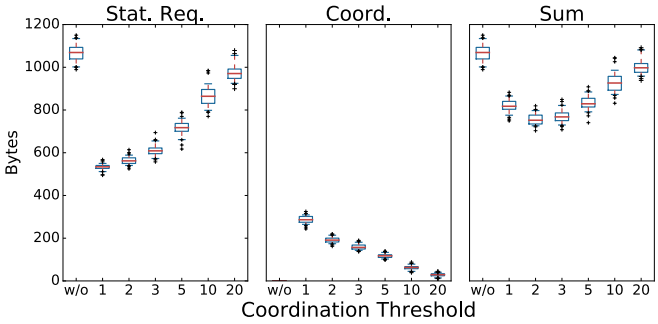


Fig. 5. Cost reduction evaluation.

B. Monitoring Cost Reduction

Figure 5 shows the main contribution of DISTTM . For the default polling period of $T = 2000\,ms$, the leftmost plot of Figure 5 shows the number of bytes used for the statistic requests and replies for one controller per flow. The number reaches its peak when DISTTM is not used (w/o). In that case, the non-collaborative approach is applied and every controller must monitor all flows it is interested in. Using DISTTM , the plot shows a significant reduction of the number of bytes for statistic requests. However, it becomes apparent that the traffic grows if the coordination threshold R is increased. This results from the fact that low coordination thresholds lead to frequent re-configurations and fast assignments of *inter-network* flows to single controllers. Consequently, redundant measurements occur fewer. For large thresholds, the occurrences of redundant measurements of flows increase, as each controller monitors all flows before they are monitored only once. Thus, as visible

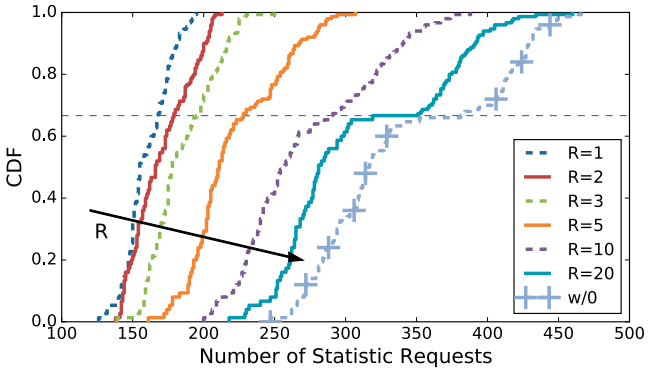


Fig. 6. Statistic request distribution among controllers.

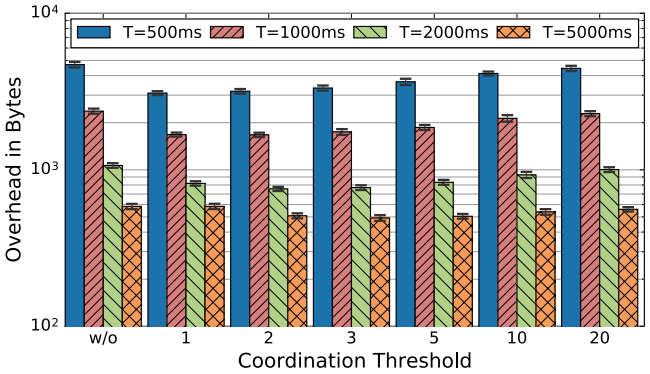


Fig. 7. Resulting traffic of the statistic requests and coordination control messages.

in the figure, the traffic from the transmitted statistic requests and replies increases as well. The plot in the middle of Figure 5 depicts the number of bytes needed for the different coordination messages. Without the use of DISTTM this value is zero since no coordination between the controllers occurs. For a small threshold R this overhead is larger, while larger thresholds, thus, rarer coordinations, lead to less coordination control messages. The resulting coordination traffic in bytes decreases with increasing threshold. Finally, the rightmost plot of Figure 5 shows the total load in terms of bytes comprising the statistic requests and replies plus the number of bytes for coordination messages. As depicted, a trade-off between the coordination threshold R and the total load exists. Coordinating on every arising flow ($R = 1$), leads to less monitoring costs (flows are always monitored only by one controller instead of each), whereas the coordination overhead is relatively high. Increasing the threshold R leads to larger monitoring costs, whereas the coordination overhead decreases. For the evaluated scenario with $T = 2000\,ms$, the operating point is between $R = 2$ and $R = 3$.

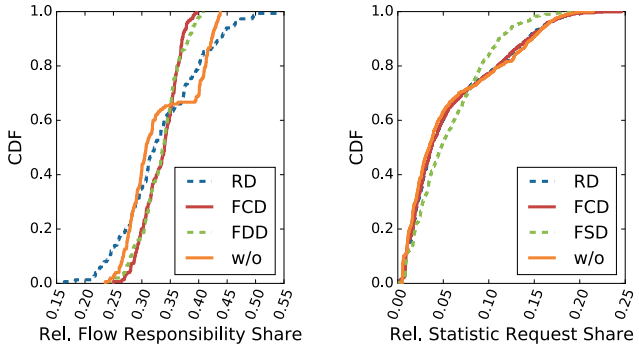
Figure 6 depicts the distribution of statistic requests per controller per flow for the different coordination threshold (R) configurations. It can be seen that the total number of requests is significantly lower using DISTTM. For $R = 1$ the system must request the least statistics. Even for high thresholds, the number is still lower than for the classical approach without DISTTM. The reduced number of statistic requests for a decreasing coordination threshold R originates from the fact the frequency of redundant, thus, unnecessary requests is reduced. As already mentioned, a lower R leads to more frequent re-configurations and fast assignment of redundantly measured flows to single controllers. In addition to the total amount of requests, the cumulative distribution function (CDF) depicts the distribution of the requests among the controllers. Regarding the distribution without the use of DISTTM, we observe that the number of statistic requests has a shift after $\frac{2}{3}$ of the data points. In the given evaluation topology (cf. Figure 4) controller C_B must handle more flows in average than C_A and C_C, as it located in between. Due to this focal position and an equal distribution of flow sources and destinations, it must monitor more flows than the other two controllers. As a result, an unfair distribution occurs. DISTTM balances the distribution more if responsibility for flows are shared between the controllers. For large thresholds, the distribution is still less balanced due to the larger number of redundantly monitored flows. With a lower coordination threshold, the negative impact of C_B's focal position nearly disappears due to frequent re-configurations and the faster unique assignment of flows to controllers. A more elaborated view on fairness and load balancing is presented in Section IV-D.

C. Impact of the Polling Period

Figure 7 depicts the overall traffic (note the logarithmic scale) comprising the statistic requests and replies plus the coordination control messages for a varying statistic polling period T and a varying coordination threshold R. As expectable, it can be observed that an increasing polling period reduces the overall traffic. However, for shorter polling periods the total savings are higher than for longer periods, whereas relative savings are comparable, although hardly identifiable in the figure. Apart from that, it is observable that the polling period influences the operating point. For a short period, e.g., $T = 500ms$, the lowest total overhead is at $R = 1$. As the period is extended, e.g., $T = 5000ms$, the operating point shifts to larger thresholds ($R \approx 3$). Hence, for short polling periods, the coordination overhead has less influence on the total load. The other way around, lower frequencies lead to larger influence of coordination control messages, thus minimizing the positive effect of DISTTM and the collaborative estimation of traffic matrices.

D. DISTTM Fairness

In order to examine the fairness, Figure 8a depicts the share of flows a controller has to monitor in relation to the total number of flows monitored by all controllers. For the use of DISTTM, three fairness schemes are applied and compared to the classical approach (w/o DISTTM). The classical approach and the *RD* scheme show rather unfair distributions. For the classical approach, C_B suffers from its central position in the topology, as shown by the shift for $\frac{1}{3}$ of all measurements. In contrast, the *FCD* scheme and the *FDD* scheme (cf.



(a) Flow responsibility distribution among controllers.

(b) Statistic request distribution among switches.

Fig. 8. Fairness evaluation results

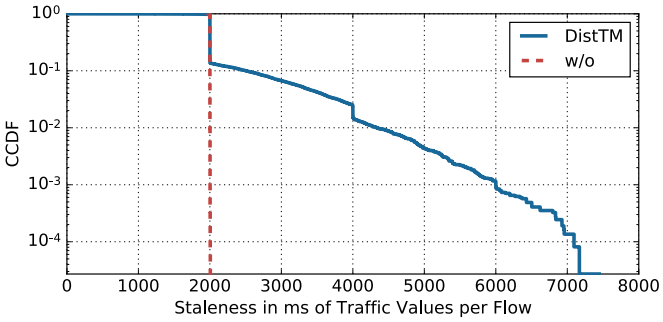


Fig. 9. Traffic matrix entry staleness.

Section III-D) lead to a steep curve indicating a balanced distribution of responsibilities among the controllers. Thus, the proposed schemes improve the load fairness on controllers. Due to the fact that *intra-network* flows occur in all three networks with the same probability, *FDD* and *FCD* behave similarly.

For the next experiment, the network of controller C_C is enlarged and consists of four times more switches. However, the probability for a flow starting or ending in one network is unchanged. Hence, the switches in the small networks may be polled more often in average when *FCD* is applied. Figure 8b depicts the portion of statistic requests in relation to all triggered requests that must be processed by a switch. It becomes apparent that the fairness in terms of load on a switch is improved using the *FSD* scheme. This results from the fact that the *FSD* scheme is the only scheme, which considers the current state of the switches.

$$f(x_1, .., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \quad (1)$$

In order to capture the fairness by a single value, we calculate the *fairness index (FI)* proposed by Jain [6]. Equation (1) produces a single value between 0 and 1, where 1 reports the highest fairness.

TABLE II
OBSERVED CONTROLLER AND SWITCH FAIRNESS.

Scheme	Controller Fairness	Switch Fairness[6]
RD	0.9510	0.5377
FCD	0.9893	0.5400
FDD	0.9907	—
FSD	—	0.6736
w/o DISTTM	0.9677	0.5413

[6]Adapted topology: the network of C_C contains four times more switches; equal probability of flow source and destination per network.

Using the unmodified scenario, the *FCD* and *FDD* schemes balance the load among the controllers in the fairest manner. Both FIs are qualitatively equal, as listed in Table II and already depicted in Figure 8a. For the modified scenario with the adapted topology the second column of Table II outlines the fairest load balance among the switches if the *FSD* scheme is applied. A fair balancing of statistic requests between the switches cannot be provided if fairness schemes are applied that ignore the different number of switches in the network as well as the current load on switches.

E. Impact of DISTTM on Performance

To examine potential trade-offs, it must be analyzed to which extent the traffic matrix estimation performance is influenced. We use the matrix entry staleness to represent the performance. For the sake of simplicity, all controllers poll statistics with the same frequency. In the classical approach, the delay between the actual values and the available data in the controller consists only of the statistic reply transmission delay between the switch and the controller. Using DISTTM, this delay is extended by two further delays. At first, the delay between a received statistic reply and the end of the measurement period, where a statistic batch message is exchanged between two controllers is added. Furthermore, the transmission delay of this statistic batch message between the controllers must be added. Since the statistic polling interval is typically much larger than the transmission delay, they can be neglected. Figure 9 depicts the CCDF of the traffic matrix entry staleness for the default statistic polling rate of $T = 2000\,ms$. It becomes apparent that about 90% of the values are updated every $2000\,ms$ (note the logarithmic scale). However, some values are staler using DISTTM. Whenever a coordination is triggered and its instructions are sent to a controller, the statistics of a flow which is assigned to another controller are not polled anymore. The next update for this flow arrives with the next statistic batch message of the responsible controller. In the meantime the affected entries are not updated. As the figure shows for a polling period of $T = 2000\,ms$, most remaining entries are updated in less than $2 \cdot 2000ms = 4000ms$. Shorter intervals are also possible during the coordination procedure. Altogether, DISTTM leads to an exponentially distributed excess staleness.

After a coordination the staleness is restored to T.

F. Summary

The results of the evaluation demonstrate that DISTTM significantly reduces monitoring costs resulting from a reduced number of required statistic requests per controller. However, introduced coordination messages lead to a trade-off between monitoring costs and additionally added overhead. We control this trade-off through a coordination threshold which controls the frequency of coordinations and improves the flow assignment. Furthermore, the results reveal that the staleness of matrix entries remains the same for approximately 90% of all measurements. Apart from that, the evaluation shows that the introduced fairness schemes allow an adjustable and fairer load distribution among controllers and switches, respectively.

V. CONCLUSIONS

In this paper we show how the collaboration between SDN controllers significantly reduces monitoring overhead while sacrificing a negligible performance fraction. The key to this reduction is a collaborative system in a distributed SDN control plane denoted DISTTM that divides the monitoring tasks such that redundant flow monitoring is eliminated. We use DISTTM to exemplarily collect estimates of traffic matrix information. We measure the performance of the monitoring system using the traffic matrix entry staleness. The empirical results show an exponentially distributed excess staleness when DISTTM is utilized. We explore the assignment of monitored flows to corresponding controllers based on different criteria. For example, we show the impact of different fairness allocations on the respective controller and switch load distributions. Note that the SDN controllers need not to be in the same domain. We keep the investigation using real world network traces and topologies for future work. Further investigations will also explore adaptive polling rates per controller and dynamic fairness schemes.

ACKNOWLEDGMENT

This work has been funded in parts by the German Research Foundation (DFG) as part of project B1, C2 and C3 within the Collaborative Research Center (CRC) 1053 – MAKI. Parts of this work have been conducted within the SENDATE project, a project funded by the German Federal Ministry of Education and Research (BMBF). The authors would like to thank Julius Rückert and Amr Rizk for their valuable input and contributions.

REFERENCES

[1] B. Awerbuch, "Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election, and Related Problems," in *ACM Annual Symposium on the Theory of Computing (STOC)*, 1987.

[2] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2014.

[3] S. Chowdhury, M. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low Cost Network Monitoring Framework for Software Defined Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[4] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, October 2004.

[5] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2012.

[6] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[7] L. Jose, M. Yu, and J. Rexford, "Online Measurement of Large Traffic Aggregates on Commodity Switches," in *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2011, pp. 1–13.

[8] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A nonstationary Poisson view of Internet traffic," in *IEEE International Conference on Computer Communications (INFOCOM)*, vol. 3, 2004, pp. 1558–1569.

[9] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A Distributed Control Platform for Large-scale Production Networks," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.

[10] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks," in *ACM Hot Topics in Networks (HotNets)*, 2010.

[11] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically Centralized?: State Distribution Trade-offs in Software Defined Networks," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2012.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[13] A. Medina, C. Fraleigh, N. Taft, S. Bhattacharyya, and C. Diot, "Taxonomy of IP traffic Matrices," in *SPIE ITCom: The Convergence of Information Technologies and Communications*. International Society for Optics and Photonics, 2002.

[14] Open Networking Fundation, "Software-Defined Networking: The New Norm for Networks," *ONF White Paper*, 2012.

[15] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[16] S. Shirali-Shahreza and Y. Ganjali, "FleXam: Flexible Sampling Extension for Monitoring and Security Applications in Openflow," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013.

[17] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A Two-Tier Resource Management Model for the Internet," in *IEEE Global Communications Conference (GLOBECOM)*, 1999.

[18] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *USENIX Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN)*, 2010.

[19] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic Matrix Estimator for OpenFlow Networks," in *Passive and Active Measurement*, A. Krishnamurthy and B. Plattner, Eds. Springer, 2010, vol. 6032, pp. 201–210.

[20] P. Tune and M. Roughan, "Internet Traffic Matrices: A Primer," in *ACM SIGCOMM eBook: Recent Advances in Networking*, H. Haddadi and O. Bonaventure, Eds., 2013, vol. 1, pp. 108–163.

[21] N. van Adrichem, C. Doerr, and F. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[22] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. Madhyastha, "FlowSense: Monitoring Network Utilization with Zero Measurement Cost," in *Passive and Active Measurement*, M. Roughan and R. Chang, Eds. Springer, 2013, vol. 7799, pp. 31–41.

[23] M. Yu, L. Jose, and R. Miao, "Software Defined Traffic Measurement with OpenSketch," in *USENIX Symposium on Network Systems Design and Implementation (NSDI)*, 2013.

[24] Y. Yu, C. Qian, and X. Li, "Distributed and Collaborative Traffic Monitoring in Software Defined Networks," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2014.

[25] Y. Zhang, "An Adaptive Flow Counting Method for Anomaly Detection in SDN," in *Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2013.

STEAN: A Storage and Transformation Engine for Advanced Networking Context

Marc Werner*, Johannes Schwandke*, Matthias Hollick*,
Oliver Hohlfeld†, Torsten Zimmermann† and Klaus Wehrle†
* Technische Universität Darmstadt, Secure Mobile Networking Lab (SEEMOO)
† RWTH Aachen University, Chair of Communication and Distributed Systems (COMSYS)

***Abstract*—Legacy Internet systems and protocols are mostly static and keep state information in silo-style storage, thus making state migration, transformation and re-use difficult. Software Defined Networking (SDN) approaches in unison with Network Functions Virtualization (NFV) allow for more flexibility, yet they are currently restricted to a limited set of state migration options. Impeding the sharing of networking and system state severely limits the ability to optimally manage resources and dynamically adapt to a desirable overall configuration. We propose a generalized way to collect, store, *transform*, and share context between NFs in both the legacy Internet and NFV/SDN-driven systems. To this end, we design and implement a *Storage and Transformation Engine for Advanced Networking Context (STEAN)*, which constitutes a shared context storage, making network state information available to other systems and protocols. Its pivotal feature is the ability to allow for state transformation as well as for persisting state to enable future reuse. By means of experimentation, we show that STEAN covers a diverse set of challenging use cases in legacy systems as well as in NFV/SDN-enabled systems.**

I. Introduction

Network management currently undergoes massive changes towards realizing a more flexible management of complex networks. Recent efforts include 1) rethinking the control plane design by applying operating system design principles to realize Software Defined Networking (SDN), and 2) Network Function Virtualization (NFV) inspired by the success of virtualization in the server market. These advances aim at a more flexible and dynamic service deployment, increased resource utilization, improved energy efficiency, vendor independence, and ,ultimately, decreased operational costs.

While these techniques advance *packet processing* and *service control*, they do not address *state management*. However, to achieve the true benefits of network and service virtualization as well as control plane programmability, scalable state sharing is of high importance—in particular when attempting to virtualize stateful network functions (NFs). This requirement has lead to the development of various systems that allow explicit state migration between NFs such as Split/Merge [1], OpenNF [2] or StatelessNF [3].

Despite their success, current state sharing mechanisms are customized solutions tailored to specific use cases and are ignoring the fact that they continue to use closed "silo style" storage as shown in Fig. 1 (a).

We advocate to *break these silos open* and allow state to be shared within the entire ecosystem of a network, ranging from

ISBN 978-3-901882-83-8

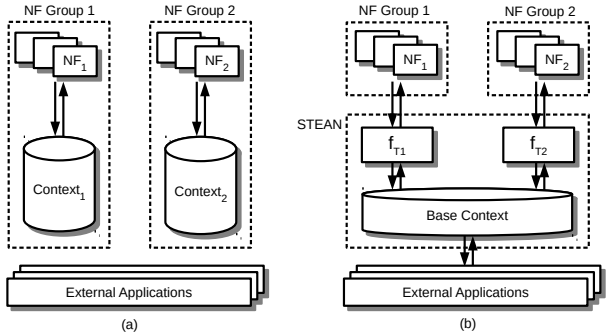


Fig. 1. Current state sharing frameworks only allow sharing between functions of the same group (a). We enable sharing context between different groups by using a common base context and mapping the function specific context using the transformation functions f_{T1} and f_{T2} (b).

SDN controllers over NFs to routers and protocol implementations (Fig. 1 (b)). By following this approach, we pave the way for realizing a *network state plane* in which network state is decoupled from the implementation of NFs similar to the decoupling of the control and data plane introduced by SDN. This decoupling of packet processing and network state will lead to a more flexible and dynamic network management, and further boost the deployment of new and innovative services.

New network management functions can use the state plane to easily aggregate state of multiple functions operating at different layers without requiring explicit support in each function. The proposed state decoupling thus enables new ways for network state cross-layering that is harder to achieve in current isolated solutions. In a similar fashion, network state can be migrated between functions more easily without explicit support.

As state migration between NFs is challenged by different state representations, we further advocate the use of *state transformation functions*. Transformation functions allow us to leave the internal state representation of the legacy systems unaltered while still sharing state with other systems. Since the transformation functions operate within the state plane rather than in each network function, we limit the explicit need for supporting state migrations. This approach surpasses the flexibility of existing solutions and enables us to generalize state management across multiple systems.

Extending the sharing and re-use of information between systems and protocols *beyond* state information further opens networks to a more flexible and dynamic management. We call

this extended set of information the *context* of a networked system. The context includes the internal state as well as the metadata describing how to interpret the stored information. Additionally, the context includes the current configuration parameters, monitoring information and historical records.

We summarize our contributions as follows:

1) We propose a *Storage and Transformation Engine for Advanced Networking Context (STEAN)*. It provides a generalized way to share context in a diverse set of core functionality such as routing, network processing, and dynamic protocol adaptation. This relies on collecting and managing context from different sources (e.g., NFs, protocols on all layers of the network stack) using their preferred state representation, thus replacing per-entity state storage with a shared context management. It makes this dynamic information available to other systems and protocols, and stores and persists the current context to re-use at later points in time.

2) We introduce transformation functions that allow for context sharing between systems that were not originally built with sharing in mind. Transformations allow STEAN to be integrated into legacy systems and to interoperate with arbitrary protocols, which permits the seamless extension of existing protocol stacks and network topologies. Furthermore, transformation functions allow us to share context between different NFs that are—until now—only designed to exchange state between instances of the same implementation.

3) We demonstrate the functionality of STEAN and evaluate its general applicability as well as its performance in two selected use cases.

II. Use Cases

We provide several motivating use cases that show why broader sharing of context is beneficial for NFs as well as for existing systems such as routing protocols in Wireless Multihop Networks (WMNs). Moreover, we show that explicit support of context sharing is essential for the development of future networks. To demonstrate the general applicability, we discuss both abstract and specific use cases.

A. UC1. Migration of Network Functions: The migration or sharing of context is a core enabler of employing virtualized NFs. This allows NFs to not only scale dynamically but keep per-flow information consistent across all instances. For example, an IDS keeps state about each flow, and rerouting the flow to a different IDS instance can significantly impact the accuracy due to missing context. Thus, context sharing improves the detection rate while still supporting dynamic scaling and flow redirection.

B. UC2. Reconfigure Network Functions: NFs are currently unable to directly share context with other systems. All information exchanged between groups of NFs flows via a central controller, limiting the available context to a predefined set that is known to the providing and consuming NF as well as to the controller. An asset management system like PRADS [4] might want to share information about hosts and services with an IDS to allow event-to-host/service correlation, or an IDS might wants to consent a firewall to access a list of malicious flows in order to block them.

Direct sharing of context adds robustness as it enables a decentralized context management, and avoids bottlenecks on the control plane when the shared context is large.

C. UC3. Switching Routing Protocols: Wireless Multihop Networks are a key technology in fifth generation (5G) wireless networks [5]. Today WMNs are deployed in environments where wired infrastructure is either not available or too expensive to deploy [6]. Currently, a variety of parameters and environmental conditions have to be considered when planning and deploying a WMN. These considerations determine a choice of technology and protocols that are fixed over the lifetime of the network as changing or adapting the networking stack to varying conditions or usage patterns basically results in deploying a completely new system configuration.

The dynamic adaptation of routing protocols provides an exit route to this dilemma. The protocol change within a WMN, however, must be seamless, without interruption of end-to-end connectivity and transparent to the end user.

III. Context Transformation

The support for *context transformations* is the core enabler for sharing context between different network components. STEAN implements transformation functions that enable connected clients to share information without agreeing on a common context. Hence, a client might profit from the information others have contributed without being explicitly aware of the existence, or even the context, of other systems or protocols.

We use a running example throughout this section to show how transformation functions can be employed to share context between independent NFs: a network consisting of a Network Address Translator (NAT), an SDN-enabled switch that balances the traffic load between two firewalls, and an IDS (Fig. 2). All NFs in this example are STEAN-enabled. The SDN controller providing rules for the switch is connected to STEAN, where it stores its state, including the SDN rules. During normal operation, the different NFs operate independently of each other.

Now, we consider the following failover scenario: Link 1 carrying the traffic assigned to Firewall A fails so all traffic is re-routed to Firewall B. The traffic load exceeds the capacity of a single firewall instance, thus traffic must either be dropped or SDN rules must be dynamically generated to enable a pre-filtering on the switch.

A. Concept

Transformations allow developers to create and use an extensible set of functions that acts as an additional layer between the client and the context storage. This layer is responsible for translating between the client-specific context and the common base context. It allows the client to store and retrieve the information “as is” and “as needed” without adapting its internal representation to the one used in the context management system. The client does not need any information on how the context of other clients has to be interpreted. This

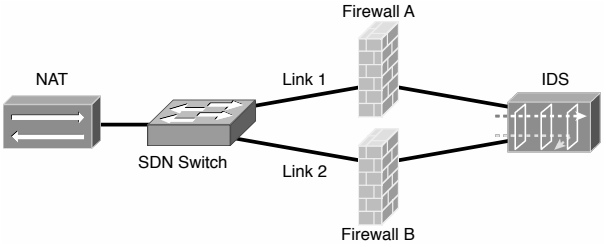


Fig. 2. Simple network to show the advantages of sharing context between different NFs. All NFs as well as the SDN controller are connected to a central STEAN instance (omitted for clarity).

interpretation is provided by the transformation functions that offer a client specific view on the base context.

In our example, the NFs can share state between each other to allow for a flexible load balancing and dynamic reconfiguration in case of failure. Each NF as well as the SDN controller keep their internal state representation, and STEAN provides transformation functions for each client. We identified four different types of transformation functions:

1) *Filter Functions* are applied during data retrieval and limit the results to the context information that is relevant to the client. For example, filters allow the NAT to only select the specific state relevant for the currently inspected packet instead of retrieving a large information base for all active translation rules.

2) *Mapping Functions* are applied to transform the client specific context to the base context and vice versa. Additionally, these functions can be used to transform serialized protocol objects within a request to the base context. This allows for minimal modifications on the client side as all mappings to the context definition are done within the context management system. In our example, the firewall as well as the SDN controller can continue to store context information using their internal state representation. In case of failure, the SDN controller is able to request additional rules from STEAN that are generated from the firewall state using mapping functions. The controller does not need to understand the state representation of the firewall but is able to use the additional information provided without adaptations.

3) *Aggregation Functions* allow for sub-context re-use. They enable the context management to combine two or more existing contexts to a single new context. Aggregations can thus be compared to the JOIN operation in traditional databases. After registration, clients can use complex queries for data retrieval in the same way they do for standard annotations. For example, the SDN rules generated from the firewall state (as described above) can be aggregated with the SDN rules stored by the controller to create a unified rule set that can be directly installed on the switch.

4) *Modifier Functions* are called on (filtered) data items retrieved from the storage. The functions can change the actual data within the item, alter the metadata attached to the entry, or modify custom metadata the client contributed. In our example, a modifier function can be used to add additional information from which state the SDN rules are generated.

Transformation of partial context, i.e., context information not providing the complete state required by a client, is explicitly supported. Partial context can occur when a new client is connected and other clients only gathered parts of the required state. When partial context is available, the client can retrieve the stored state information using transformation functions to convert the context but has to gather the missing information using system or protocol specific mechanisms. Then, the additional context can be contributed to the context management system and, hence, made available to other clients. For instance, one routing protocol might only be able to contribute one-hop neighbors to the context storage while another protocol also requires all two-hop neighbors of a node. When switching protocols, the latter can retrieve the list of one-hop neighbors from context management and start the discovery of two-hop neighbors based on this information.

STEAN-side transformations allow us to make use of a shared cache between clients when they connect in parallel and query the same context information. This cache reduces the load on the system and thus decreases the response time for subsequent requests. Additionally, we are able to reduce the communication overhead between STEAN and the clients when filtered or aggregated context information is requested as only the needed set of context information is returned to the client. The firewalls in our example use the same state representation and thus share a common cache. This results in faster access times when packets matched by the same rule are processed on either instance.

B. Features

STEAN allows clients to specify the transformation functions between their context and the base context upon connection. Those functions are then called each time data a client reads or writes data, and the base context is automatically mapped to the client specific context and vice versa.

The mapping does not need to be a static function but can be adaptive to runtime configuration changes. This allows the client to dynamically adapt its context to the current environment without the need for redefining annotations or exchanging the transformation function. In our example above, the IDS can dynamically adapt the information retrieved from STEAN when a suspicious flow is detected and extend the number of evaluated flow properties without reconfiguration. This allows to faster detect attackers by looking for flow context stored in STEAN—that is contributed by the firewall systems—once a suspicious flow is identified.

Transformation functions are designed to be modular and composable: functions can call other functions to create complex transformations with minimal effort. Additionally, transformation functions query external systems to retrieve additional information. For example, the transformation between an IP address and a MAC address requires to issue an ARP request on the local network.

C. Limitations

Transformation functions are mainly limited by their complexity and the resulting loss in performance. The complexity of a transformation not only depends on the function itself but also on the design of the base context. If the base context efficiently supports the envisioned clients and thus the needed transformations, the overhead can be kept minimal and the performance loss is mostly negligible.

Furthermore, the client developer has to manually define the required transformation functions. Currently, there is no automatic system that generates transformation functions from either existing implicit context representations within the client (data structures, object relations), or from an explicit description of the client specific context (annotations, models). Naturally, state transformations are further bound by the available state. That is, state can only be transformed but not inferred. For example, transforming state from a routing protocol maintaining a 1-hop neighborhood to a 2-hop neighborhood is only partially possible, as the missing state needs to be inferred by the protocol itself.

D. Designing Transformation Functions

When designing and implementing new transformations, it is important to keep the computational overhead as low as possible since all information stored in and retrieved from STEAN potentially passes the functions. Moreover, it is necessary to evaluate the cost of using transformations against the cost of locally retrieving or calculating the information within the client without accessing the context management system. In some cases, it might be more efficient to (re-)generate the context in the client rather than extracting the needed context from STEAN using a complex transformation function. This is especially true for information with a short lifetime which requires regular updates that prevent efficient caching of transformation results.

As the complexity of the transformation functions depends on the design of the base context, a close interaction while building the base context and the transformation functions might reduce computational overhead. This includes that transformations should target a small scope of the overall context and apply filter functions as early and as restrictive as possible. Restrictive filtering limits the number of data items processed by other, potentially more complex, functions to a minimum, thus improving the response time of STEAN. This also contemplates that functions exit as early as possible: if the NAT in our example requests a single state item, the filter function must be terminated after the item is found.

During a lookup operation, transformation functions should be called in a specific order: 1) filter functions reduce the amount of data retrieved from storage, 2) mapping functions translate the base context to the client-specific context, and 3) aggregate functions then unify different data items to provide a single context to the client.

While technically feasible, transformation functions should not fetch information from external sources unless this information is a direct transformation of stored context. Additional functionality should be placed within the client as it is a feature of the implemented system or protocol rather than a necessity of sharing context. In general, transformation functions should not generate new state but work on the existing context stored by the clients. Additionally, in order to support concurrent access, all transformation functions must not directly alter the stored data but only transform the information received from the storage subsystem.

IV. System Design

STEAN is designed as a node-local system that manages all context information of connected clients. A *node* is not limited to a physical system but can be any network entity with a well-defined purpose. This can be instances of a virtualized NF that form a cluster of Intrusion Detection Systems (UC1), or a single wireless device that is forwarding traffic in a WMN (UC3). A certain number of clients thus form a node. The design is centered around the transformation functions as the enabler for a generalized context management system.

A. Components

STEAN consists of five core components which are assigned specific tasks within our architecture and can be exchanged with other implementations. Fig. 3 gives an overview of the components and their interaction.

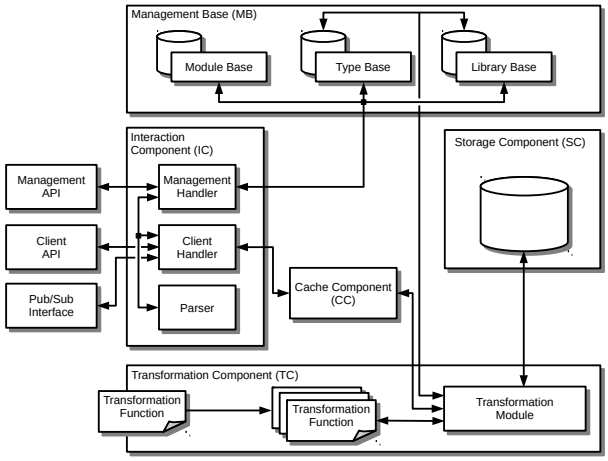


Fig. 3. Architectural overview of STEAN. The arrows show the interaction between components.

1) Interaction Component: The Interaction Component (IC) is responsible for handling incoming commands. These commands can be either context requests from a client, or updates to the state of STEAN itself such as adding new transformation functions or registering additional annotations.

2) Storage Component: The Storage Component (SC) holds the actual data that the clients add and query. Data within the SC is grouped by annotations and organized in several sets. These sets are used for efficient data retrieval since only those sets using the requested annotations have to be searched. In addition, metadata is attached to each data item. This metadata can either be provided by the client, by transformation functions that are called during the storage process, or by STEAN itself.

3) Transformation Component: The Transformation Component (TC) implements transformation functions as described in Section III. The TC is invoked on every query and connects to the SC. The TC either transforms the data retrieved by the SC to match the context of the requesting client (lookup request) or transforms the inserted data to the base context (add or modify request).

4) Cache Component: To be better suitable for performance critical NFs, STEAN makes intensive use of caching. The Cache Component (CC) is placed between IC and TC. The cache thus holds context information where the client specific transformation functions are already applied. The placement keeps the computational overhead of applying transformation functions as low as possible but leads to a minimal reuse of cached results across clients.

We opted against a shared cache placed between TC and IC but for a cache holding an individual set of results for each client. The diversity of clients would not allow for a wide reuse of cached entries as each client specifies its own context. A shared cache instead extend the number of entries per cache set and thus lead to a higher retrieval time. Additionally, we decided not to place a shared cache between SC and TC. This placement would allow for a higher reuse of cached information but the gains are much lower since transformations have to be applied to each returned result.

The cache allows to reduce the retrieval costs for state lookups, which is relevant for performance critical NFs (e.g., functions performing per-packet lookups at line rate). However, such high-speed NFs are out the of scope of this paper, and we leave their evaluation for future work.

5) Management Base: The complete metadata and the state of STEAN itself is represented in the Management Base (MB). The *Type Base* within the MB stores information about known annotations and possible attributes, while the *Library Base* manages the transformation functions available. The *Module Base* subcomponent holds a list of clients, and their registered annotations and transformation functions.

B. Communication and Interaction

STEAN provides two interfaces for outside communication. The Client API is used by the accessing systems and protocols to store and retrieve context, and the Management API is used to control the behavior of STEAN itself. While the first interface is openly available to all clients, the second interface is protected to prevent unauthorized reconfiguration.

1) Client API: STEAN supports multiple annotation sets that are registered by clients. Upon connection, each client has to register and provide the annotation (sub-)set it will use, and specify the transformation functions to convert the client-specific context to the base context and vice versa. After successful registration, the client can access the specified annotations and transformation rules while access to other annotations or transformations is denied. This initial registration forces each client to completely model its environment and describe its context compared to the base context before access is granted. Changes to the set of annotations or transformation functions require a full re-connect of the client.

STEAN also offers a publish-subscribe interface that notifies connected protocols when changes to subscribed annotations occur. This interface can notify connected clients such as monitoring systems when the stored context changes.

2) Management API: The management interface provides methods to alter the base context of the service, add and remove annotations, and register new transformation functions. Access limitations on the interface prevent clients from registering arbitrary annotations or transformation functions that have no value to other clients (as they are unknown), or even compromise the service itself as malicious functions might leak sensitive data.

V. Implementation

STEAN is implemented in C++ and runs as an independent service on the host system. We successfully tested the functionality on Linux, FreeBSD and Apple OS X.

A. Storage System

The storage system is implemented on top of a XML database using RapidXML [7], and the items in the database are accessible via a management plane. The management plane is implemented as a map of pointers that allows for direct access to the requested annotation and handles the lifetime of each data item. The database consists of several sets, one per available annotation. Each data item can currently only be tagged with one annotation and is thus associated to exactly one set. To remove this limitation, the management plane provides additional indices that allow for direct access across annotation sets. These indices can be seen as virtual annotations and can be accessed in the same way.

STEAN also supports a *snapshot* feature that can be used to create a persistent copy of the stored information and the current state of the service. The snapshots, however, do not contain the shared libraries registered but assume that the libraries are available at the same location.

B. Function Libraries

Transformations are implemented as Unix shared objects and have to be loaded via the Management API. This enables us to add functions on demand without shutting down or even recompiling STEAN. After registering the library system wide, each client needs to register the used transformation functions together with the base context annotation and the mapping annotation within its client context. This ensures that STEAN calls the correct transformation function when an annotation is requested without the need to specify the function on each request, and prevents inconsistent mappings between requests from the same client. Additionally, it keeps the size of request messages low and thus increases the response time of STEAN.

C. Client Implementation

We designed and implemented a client library that provides convenient access to STEAN without the need for the client developer to handle the connection management and the XML message building and parsing.

1) UC1. Migration of Network Functions: The PRADS asset monitor [4] is a passive network monitor that allows to map the services running in a network and detect changes in real time. It uses TCP and UDP fingerprinting to identify operating systems and service applications. PRADS also keeps an internal state table to identify flows in the network and provide information on the services offered and used by the networked systems.

The STEAN support for PRADS is built on top of the OpenNF [2] modifications that allow to migrate NF state between different instances. Instead of migrating the state via the controller, we directly share context information between the PRADS instances using STEAN. We therefore modified PRADS to be a STEAN client while still supporting the OpenNF controller messages to initiate the migration of flows.

The PRADS instances share the complete internal state of all observed flows using STEAN. Beside a unique identifier per flow, the protocol 5-tuple and the IP protocol version, the flow state also includes timestamps for the first and last seen packets, the number of packets observed for the flow, as well as the total size of transmitted data for each direction. PRADS also includes the hardware protocol and any TCP flags observed into the flow state along with a list of identified assets for source and destination.

As we are only sharing information between instances running the same implementation, the only transformation functions required are filters to select specific flow entries based on the unique flow identifier assigned by PRADS.

2) UC3. Switching Routing Protocols: We modified implementations of the Ad hoc On-Demand Distance Vector (AODV) routing protocol [8] as well as the Optimized Link State Routing (OLSR) protocol [9] to support UC3. The protocols are implemented using the Click Modular Router [10] framework and we extended the state handling elements to connect to STEAN. Each protocol uses a special Click element that is responsible for specifying the protocol context, and registering annotations and transformation functions during system startup. This element also handles the communication with STEAN during protocol operation.

Accompanying the implementation changes, we designed a base context that closely matches the requirements of the routing protocols. Specifically, we share the list of one- and two-hop neighbors as well as the list of multipoint relays and the routing tables. The protocols do not hold any local context but solely access information stored in STEAN.

We have implemented transformation functions for both routing protocols that 1) filter entries in the routing table to select only the specific route for a single packet and 2) map the format of an entry to match the internal format of the accessing routing protocol.

VI. Evaluation

We evaluate STEAN in the use cases UC1 and UC3 from Section II since they represent the diversity of possible operation scenarios for a context management system. Before we present the results from the use case study, we show the general applicability of STEAN and how the usage of transformation functions influences the system behavior.

A. General applicability: Our goal is to understand the performance of basic STEAN operations. We evaluate the behavior of STEAN using a simple client that is able to store and retrieve context information. The client inserts and reads IPv4 addresses that are either represented as a string with dots separating the octets or each octet represented as an integer value. Additionally, transformation functions are available in STEAN to convert between these two formats. The evaluation is conducted on a single machine with a Quad-core Intel Xeon CPU and 16 GB of memory. All caches are disabled to show the raw performance of the transformation engine and the context storage.

Fig. 4 shows the time per insert for inserting 1000 (a, d), 10.000 (b, e) and 100.000 (c, f) unique IPv4 addresses both with and without applying the transformation function.

Fig. 4. Time per insert without calling a transformation function (a–c), and with calling a transformation function to convert the representation (d–f).

Our results show that writing to STEAN takes constant time regardless of the number of entries already stored. Saving one context entry takes about 140 μs when no transformation function is employed and around 180 μs when the simple function described above is used to convert the representation. Additionally, we see that at least $1/3$ of the request completion time is spent on socket communicatin. The time shown for transformations, even when no function is executed, is due to the overhead passing all requests through the transformation engine and not interfacing with the storage directly. While we focus on a single client in Fig. 4, we remark that additional clients have a negligible performance impact and only increase the variability of the insert time (not shown).

Fig. 5 depicts the results for reading one out of the 1000 (a, d), 10.000 (b, e) and 100.000 (c, f) addresses inserted before. The address is selected by applying a filter function for a random but fixed address per operation.

The experiments show that the time for retrieving context is linear to the number of entries stored in STEAN. This is due to the current implementation of the storage component that is iterating over all entries for an annotation until a match is

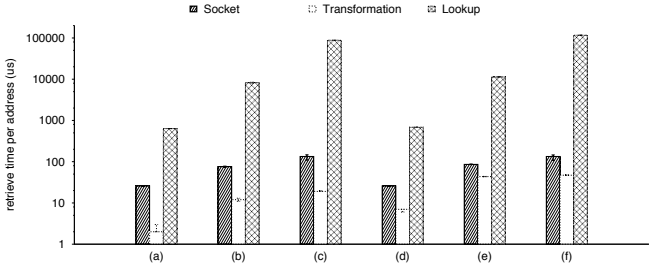


Fig. 5. Time to retrieve an address without calling a transformation function (a–c), and with calling a transformation function (d–f). The boxes represent the median and the error bars show the first and third quartile.

found. This behavior is also represented in the timings for the transformation engine as they include the time for applying the filter function. Each item is passed through the filter to check for a match and thus the transformation time also increases with the number of entries. Concurrent lookup requests do not influence the performance of STEAN as read operations are executed in parallel.

B. UC1. Migration of Network Functions: Our goal is to compare context sharing using STEAN with current state migration systems for virtualized NFs such as OpenNF. We are using a modified implementation of the PRADS asset monitor that supports OpenNF and also includes our extensions for STEAN support as described in Section V-C1. All experiments were conducted inside a Mininet [11] instance.

The data network consists of two PRADS instances ($PRADS_1$ and $PRADS_2$) that are connected to an Open vSwitch, and a dedicated host in the data network replays a university-to-cloud trace. The trace has an overall duration of approx. 20 h and contains 70 k TCP flows, $2/3$ of which are HTTP(S) flows. On average, a flow has a duration of 35 s and 33.6 flows are active in parallel. For 13 % of the time, more than 100 flows are active in parallel.

The PRADS instances are connected to STEAN using a dedicated management network that also hosts the NF controller. The controller is responsible for initiating the migration of flows between the two PRADS instances and for reconfiguring the SDN switch during migration. The setup is depicted in Fig. 6. The experiments are run on a single machine with a Quad-core Intel Xeon CPU and 16 GB of RAM.

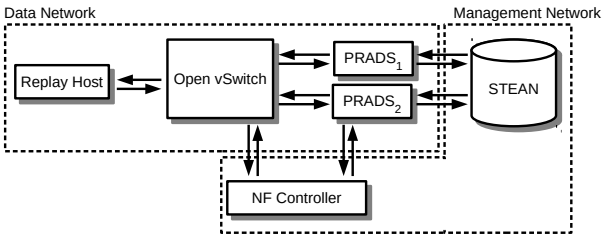


Fig. 6. Experimental setup for the evaluation of UC1.

We replay the trace at 500 packets per second and initially send all traffic to $PRADS_1$. Once it has created state for 250 and 400 flows, respectively, we initiate the migration of the flow state to $PRADS_2$.

The state is either migrated via the controller (OpenNF) or by sharing the current context using STEAN and only signalling the migration via the controller. All migrations are executed with order preserving enabled and STEAN executes filter transformations to select the context of the flow that is currently migrated. Fig. 7 shows the migration time for one TCP flow, comparing the OpenNF implementation to STEAN.

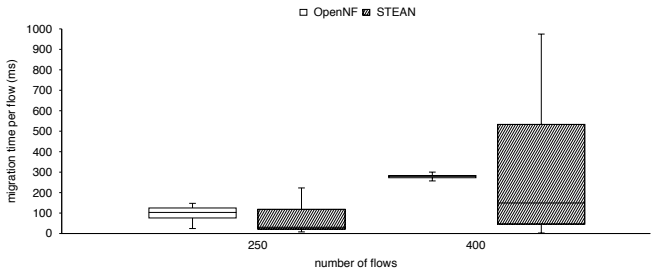


Fig. 7. Total migration time per flow context between two PRADS instances.

We observe that employing STEAN for context migration reduces the median migration time by 60 % per flow from 280 ms to 160 ms for the 400 flow case. This reduction, however, comes with an increased variability that is due to database locking of the current STEAN storage backend on inserts, delaying some concurrent requests.

Increasing the number of flows to be migrated above 400 results in a large increase in time between storing the context on $PRADS_1$ and retrieving the context on $PRADS_2$, while the times for operations involving STEAN remain almost constant as shown in Fig. 8. The overall performance decrease

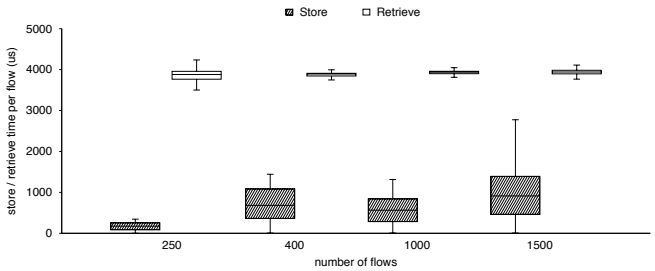


Fig. 8. Store and retrieve time per flow for migrating flows using STEAN.

when more than 400 flows are migrated is therefore not due to a bottleneck in STEAN but originates from either a congestion in the management network, or an overload of the controller.

Our results show that sharing context using STEAN is faster than migrating state employing OpenNF for the use case UC1, proving that STEAN can compete with state-of-the-art systems for migrating NF state.

C. UC3. Switching Routing Protocols: The main objective of this experiment is to demonstrate the applicability of transformation functions by providing seamless transitions between two routing protocols using STEAN.

We use a wireless mesh testbed in an office environment to conduct our experiments. Each host runs an instance of the modified AODV and OLSR implementations along with a STEAN instance to manage context. The STEAN instance

is configured with the base context and the transformation functions described in Section V-C2.

The experiments are conducted with a constant packet rate of 250 packets per second and a packet size of 1000 Bytes. The performance of STEAN does not depend on the absolute throughput of the network but rather on the packet rate as the number of requests to STEAN does not increase when using larger packets. We enabled client-side as well as STEAN-side caches for optimal forwarding performance.

First, we evaluate the behavior of OLSR when running the original implementation as well as our modifications that enable context sharing. Running OLSR with STEAN support to manage the protocol context increases the average end-to-end delay from 6.76 ms to 11.93 ms and the jitter from 1.76 ms to 103.08 ms. However, this increase is still acceptable for almost all applications running across a wireless mesh network and even allows for Voice over IP calls [12]. Here we observed that for $2/3$ of all packets the forwarding time is equal for both the standard and the STEAN-enabled implementation. The higher delay for other packets is due to blocking updates of the routing table that include packet counters and are thus altered regularly. Additionally, 95 % of all packets arrive within 33 ms and the high jitter comes from a few outliers.

Next, we execute a routing protocol transition from OLSR to AODV during runtime. The transition is triggered using a central controller and an out-of-band connection to each host as described in [13]. In Fig. 9, we show that the transition is executed without interruption in packet forwarding. The only visible effect is that the jitter is reduced and thus a better overall network performance is achieved.

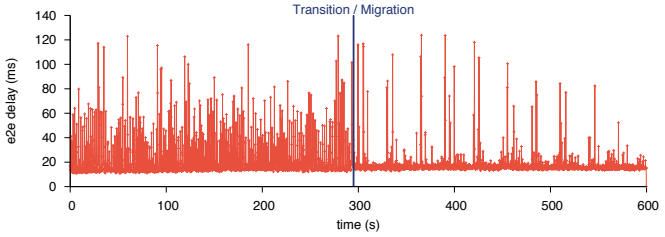


Fig. 9. End-to-end delay when migrating from OLSR to AODV during normal network operation. The transition is executed after 295 s (dark blue line).

We conclude from the experiments shown above that a common state store in conjunction with transformation functions as implemented by STEAN is able to support seamless protocol transition in WMNs with minimal overhead and enables protocol transitions during run-time without loosing end-to-end connectivity. The transition is transparent to end systems as well as overlying protocols.

D. Implementation Overhead: To quantify the modification required to support STEAN in the aforementioned systems, we counted the Lines of Code (LoC) that were added or changed in each implementation (Table I).

The results show that systems designed to share context information only require minimal changes to support STEAN. While the number of LoCs for AODV and OLSR might indicate rather dramatic changes, the actual implementation overhead was minimal since only a few functions needed to be changed. As these functions were largely scattered over the code, they increased the overall LoC count.

TABLE I
ADDITIONAL OR CHANGED CODE TO IMPLEMENT STEAN SUPPORT.

Implementation	LoC added/changed	Change in Code
AODV	542	21.4 %
OLSR	1289	49.9 %
Common Click Code	1243	n/a
PRADS w. OpenNF	144	0.7 %
STEAN shared library	972	n/a

VII. Related Work

One approach of existing work to migrate NFs is moving the complete virtual machine (VM) as done by Remus [14]. This guarantees seamless failure recovery without modifying the function itself. However, migrating the VM comes at a significant cost as not only the relevant state of the NF itself but the state of the complete operating system is transferred to the backup system on a regular basis. Depending on the load of the actual VM, this *checkpointing* can cause a significant amount of additional latency to the normal operation. Alternatively, lightweight VMs such as specialized single-process containers can be used to reduce the overall replication overhead as shown by Tardigrade [15].

While migrating the complete (lightweight) VM suffices in a failover scenario, the systems lack an efficient measure to scale on different loads as the complete state of all flows needs to be duplicated, which not only wastes memory but also might result in a false behavior of the replicated NF.

Split/Merge [1] and Pico Replication [16] provide a framework to copy, migrate and replicate the state of NFs. They allow the migration of state from several instances of the same NF when creating or destroying a copy, or in the case of failover. In addition to the above, OpenNF [2] provides coordinated control of forwarding state in SDN to avoid packet loss or re-ordering, which can lead to a degraded performance of NFs. Kothandaraman et al. [17] as well as Gember-Jacobson and Akella [18] improve the performance of OpenNF by exchanging the function state directly without involving the control plane during migration. In contrast, Kablan et al. [3] propose to keep the state externally, leaving the NF itself stateless and centralizing all state management.

Statesman [19] introduces a network-wide state management architecture that is tailored towards data centers. It focuses on the collection and migration of states from multiple network management applications. The goal is to manage the configuration state of the complete network and to allow for a coordinated network-wide state transition, while keeping track of network invariants and offering several mechanisms for conflict resolving during state migration. Statesman focuses on the network-wide configuration state of management applications, but is not designed to handle the state of protocols or NFs.

TABLE II
OVERVIEW OF THE RELATED WORK.

	Scope		State Exchange		Features			Use Case		
	VM	App.	Direct Sharing	Migration	Persistence	Transformations	Decoupled State	NFs	Mgmt.	WMN
Remus [14]	++	−−	o	−	−−	−−	−−	+	o	−−
Tardigate [15]	++	−−	−−	+	−−	−−	−−	+	o	−−
Split/Merge [1]	−−	++	−−	+	−−	−−	−−	++	−	−−
Pico Replication [16]	−−	++	+	o	−−	−−	−−	++	−	−−
OpenNF [2]	−−	++	−−	+	o	o	−−	++	−	−−
DiST [17]	−−	++	−	++	−−	−−	−−	++	o	−−
p2p OpenNF [18]	−−	++	−	++	−−	−−	−−	++	o	−−
Stateless NF [3]	o	++	++	−	o	−−	++	++	o	−−
Statesman [19]	−−	++	+	−−	++	−	++	−	++	−−
STEAN	+	++	++	o	o	++	+	+	o	++

Table II gives an overview of the related work discussed above. It specifically shows that existing solutions focus on the migration of either the complete state of a VM (Remus, Tardigate) or the state of the NF running within this machine (Split/Merge, Pico Replication, OpenNF, Statesman).

STEAN separates the context from the actual functionality and provides a backend store for state information. While Stateless NF follows the same approach, our solution is also capable of storing state information from different protocols and applications across the network stack in a common base context and is thus capable of sharing the complete virtual machine state if required.

The current solutions provide state migration between systems of the exact same type as they directly extract the state from within the NF (Split/Merge, OpenNF, Stateless NF) or even require connecting applications to adopt to the state model of the management system (Statesman). To overcome this limitation, STEAN uses transformations to allow clients to specify their context and share information across implementations without adapting to a specific state model.

Furthermore, the existing systems focus on a single use case while STEAN specifically targets the complete network environment and provides a generalized solution for managing context information.

VIII. CONCLUSION

Sharing context information across components is essential for a more flexible and dynamic network management, and further boosts the deployment of new and innovative services. With this, we are able to overcome the limitations of current network functions and to include legacy systems such as routing protocols into new network architectures. Transformations are a core enabler for this extensive sharing as they allow us to support a large variety of network components and protocols without the need to adapt the internal state of these systems but with minimal changes to existing implementations.

We presented STEAN, a Storage and Transformation Engine for Advanced Network Context, that enables us to not only share state between instances of the same implementation but to extend the sharing of networking context beyond these boundaries. STEAN supports transformation functions by design and the architecture is centered around this core feature. Our evaluation shows that we are able to support a wide range of use cases with an acceptable overhead.

ACKNOWLEDGMENT

We thank our shepherd Olivier Bonaventure and the anonymous reviewers for their insightful comments and suggestions. This work has been co-funded by the DFG as part of the CRC 1053 MAKI and by LOEWE CASED.

REFERENCES

[1] S. Rajagopalan *et al.*, "Split/Merge: System Support for Elastic Execution in Virtual Middleboxes," in *NSDI*, 2013.
[2] A. Gember-Jacobson *et al.*, "OpenNF: Enabling Innovation in Network Function Control," in *SIGCOMM*, 2014.
[3] M. Kablan *et al.*, "Stateless Network Functions," in *HotMiddlebox*, 2015.
[4] E. Fjellskål, "Passive Real-time Asset Detection System," http://gamelinux.github.io/prads/.
[5] A. Osseiran *et al.*, "The Foundation of the Mobile and Wireless Communications System for 2020 and Beyond: Challenges, Enablers and Technology Solutions," in *VTC Spring*, 2013.
[6] M. Afanasyev *et al.*, "Analysis of a Mixed-Use Urban WiFi Network: When Metropolitan becomes Neapolitan," in *IMC*, 2008.
[7] M. Kalicinski, "RapidXML," http://rapidxml.sourceforge.net/.
[8] C. Perkins, E. Belding-Royer, and S. Das, "RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing," IETF, RFC, 2003.
[9] P. Jacquet and T. Clausen, "RFC 3626: Optimized Link State Routing Protocol (OLSR)," IETF, RFC, 2003.
[10] E. Kohler *et al.*, "The Click Modular Router," *ACM TOCS*, vol. 18, no. 3, 2000.
[11] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in *HotNets*, 2010.
[12] M. Karam and F. Tobagi, "Analysis of the Delay and Jitter of Voice Traffic Over the Internet," in *INFOCOM*, 2001.
[13] M. Werner *et al.*, "A Blueprint for Switching Between Secure Routing Protocols in Wireless Multihop Networks," in *WoWMoM*, 2013.
[14] B. Cully *et al.*, "Remus: High Availability via Asynchronous Virtual Machine Replication," in *NSDI*, 2008.
[15] J. R. Lorch *et al.*, "Tardigrade: Leveraging Lightweight Virtual Machines to Easily and Efficiently Construct Fault-Tolerant Services," in *NSDI*, 2015.
[16] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico Replication: A High Availability Framework for Middleboxes," in *SOCC*, 2013.
[17] B. Kothandaraman, M. Du, and P. Sköldström, "Centrally Controlled Distributed VNF State Management," in *HotMiddlebox*, 2015.
[18] A. Gember-Jacobson and A. Akella, "Improving the Safety, Scalability, and Efficiency of Network Function State Transfers," in *HotMiddlebox*, 2015.
[19] P. Sun *et al.*, "A Network-state Management Service," in *SIGCOMM*, 2014.

Efficient Virtual Network Isolation in Multi-Tenant Data Centers on Commodity Ethernet Switches

Heitor Moraes Marcos A. M. Vieira Ítalo Cunha Dorgival Guedes
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
Email: {motta, mmvieira, cunha, dorgival}@dcc.ufmg.br

***Abstract*—Infrastructure-as-a-Service providers need to provision and isolate their tenants's virtual networks. Current network isolation solutions either suffer from limited scalability, incur encapsulation overheads, or require advanced (e.g., OpenFlow) hardware switches. We propose LANES, a system that provides isolation between billions of virtual machines using commodity Ethernet switches without encapsulation overheads. LANES virtualizes each tenant's network address space and configures rules on each server to translate (tenant) virtual addresses to (infrastructure) physical addresses. Virtual address spaces give tenants flexibility when configuring their virtual networks, and physical addresses reduce demand on infrastructure switches. We implement LANES in OpenStack, leveraging OpenStack's network description functionalities and using OpenFlow to configure Open vSwitch on infrastructure servers. Our evaluation shows LANES ensures network isolation with acceptable rule configuration latency.**

I. Introduction

Infrastructure as a Service (IaaS) providers have a growing demand for solutions to allocate and manage the resources offered to their customers (usually called tenants) [1]. Each tenant requires network resources to interconnect a set of virtual machines (VMs) in an arbitrary topology. Ideally, except for specific interconnection agreements, traffic from one tenant's VMs should never be visible to other tenants's VMs; conversely, only that tenant's traffic should be able to reach his VMs. IaaS providers must provision network resources to guarantee isolation between customer networks.

Simple solutions to provision and isolate tenant networks, like Ethernet VLANs, do not scale to large datacenters [2]. Researchers, standards bodies, and industry have proposed several scalable alternatives over the last few years. One common approach is to virtualize the networking infrastructure using tunneling [3], [4], which incurs encapsulation overhead, or to use advanced (e.g., OpenFlow) hardware switches, which results in additional costs.

Our goal is to provide efficient traffic isolation between tenants in a datacenter environment on commodity Ethernet switches. Since each tenant is oblivious to other tenants, tenants's VMs might use incompatible and overlapping network configurations (e.g., using the same IP address). It is the responsibility of the isolation solution to handle this situation and properly deliver packets to the correct VMs.

To achieve this goal we present LANES, a platform to provision virtual networks and ensure traffic isolation on multi-tenant datacenters based on the paradigm of Software Defined Networks (SDN) [5]. LANES provides flexibility and extensibility through standard APIs.

LANES allows IaaS tenants to specify their (layer-2) network topology using OpenStack's network description language [6]. The LANES SDN controller then uses OpenFlow [7] to configure an Open vSwitch instance on each infrastructure server to provision and isolate tenant networks. LANES was designed to be compatible with existing datacenter infrastructures. LANES requires no modification to physical switches; in particular, LANES runs Open vSwitch on infrastructure servers but does not require OpenFlow-enabled network switches.

LANES virtualizes network address spaces and isolates virtual networks using packet rewriting, and does not incur encapsulation overhead. LANES uses a pair of OpenFlow rules at each server's Open vSwitch instance for each virtual link between communicating virtual machines terminating at that server. LANES's address virtualization uses only the physical servers' MAC addresses in the physical network, which avoids scalability issues associated with large layer-2 address domains that arise when virtual machine MAC addresses traverse the physical switches [8]. LANES provides flexibility to IaaS customers and scales to large datacenters while incurring minimum additional costs.

We evaluate our prototype and show that LANES induces negligible additional latency when translating packets to virtualize addresses, and that Open vSwitch rule configuration, which happens only once for each VM pair, takes less than 200ms. We also show that VM traffic under LANES continues responsive and can achieve full bandwidth utilization even when under DoS attacks.

The remainder of this paper is organized as follows. Sec. II presents the idea behind LANES and describes the packet rewriting technique used, while Sec. III discusses the implementation details of our prototype using OpenStack. Sec. IV evaluates the performance of our LANES prototype. Finally, Sec. V discusses related work and Sec. VI presents final remarks and discusses possible future work.

II. LANES

LANES provisions and isolates layer-2 virtual networks in multi-tenant datacenters. A virtual network interconnects virtual machines (VM) that run on multiple hosts. LANES runs a virtual switch on all datacenter servers to intercept packets before they are forwarded. Interception allows LANES to rewrite packets to virtualize network addresses and isolate virtual networks (Sec. II-A). We also present the algorithms

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

LANES uses to configure packet rewriting and packet forwarding (Sec. II-B). LANES requires no modification to hosted VMs, and puts no restrictions on what IP addresses VMs can use. LANES also works on unmodified commodity learning Ethernet switches and scales to datacenters with millions of VMs and hundreds of thousands of servers. We describe how we implement LANES's address virtualization and packet forwarding in OpenStack on top of existing standards using Open vSwitch, POX, OpenFlow, and Neutron in Sec. III.

A. Network address virtualization using packet rewriting

Current Ethernet switches have tens of thousands of entries in their MAC forwarding tables.[1] A few hundred servers can host tens of thousands of VMs and put significant pressure on switch forwarding tables, causing severe network performance degradation. In an unmanaged Ethernet segment, VMs can spoof Ethernet MAC addresses to perform DoS attacks on switch forwarding tables or sniff traffic.

LANES assigns a unique MAC address to each server and VM in the infrastructure. We denote the MAC address of a server (or VM) x as M_x. When VM u in server s sends a packet to VM w in server r, the virtual switch at server s rewrites the original packet changing M_u to M_s and M_w to M_r. This rewriting allows packets to traverse (from source to destination servers) a physical network made of commodity learning Ethernet switches requiring a single forwarding table entry per server in the infrastructure. No VM MAC ever reaches the network infrastructure, preventing VMs from spoofing MAC addresses. LANES requires a few KB of memory at each server to store mappings from VMs (attached to virtual networks that span that server) to servers and mappings from servers to MAC addresses.

VMs may use multiple and arbitrary IP addresses. If a server hosts VMs with identical (conflicting) IP addresses, the virtual switch needs additional information to forward the packet to the correct destination VM.

LANES associates one flow identifier F_{uw} to the traffic between each pair (u, w) of communicating IPs. Flow IDs are generated on demand when VMs start communicating and need be unique only between pairs of servers (different pairs of servers may use the same flow IDs). As Ethernet switches forward packets based on MAC addresses alone, LANES uses the source and destination IP addresses to store flow identifiers. Although the number of possible flows is up to 2^{64}, servers need to store only one flow ID per pair of communicating IPs. We note LANES creates *one* flow identifier for *all* connections between a pair of communicating IP addresses. This reduces the cases where a new identifier has to be created to when a pair of IP addresses exchange their first packet.

As an example of LANES's address virtualization, consider the deployment scenario in Fig. 1 with two virtual networks, two servers, and six VMs. We consider that servers are configured with address mappings shown in Tab. I (we explain how LANES generates mappings in Sec. II-B) and that VM i uses a single IP address denoted by A_i.

When VM u transmits a packet to VM w, server s's virtual switch intercepts a packet $P = [M_u M_w \mid A_u A_w]$. As this is an outbound packet,[2] LANES looks up the MAC address of VM w's server (this table is not shown), matches packet P's header against line 1 in Tab. I, and rewrites P into $P' = [M_s M_r \mid F_{uw}]$. When the packet enters the physical network, switches learn which port to use to reach M_s and switches that do not yet have a entry for M_r in their forwarding tables broadcast the packet. When server r receives packet P', it identifies this as an inbound packet and looks up F_{uw} in Tab. I to rewrite P' back into P and forward the original packet to VM w.

[1] A Cisco Catalyst 4500-X switch has 55K MAC forwarding table entries.

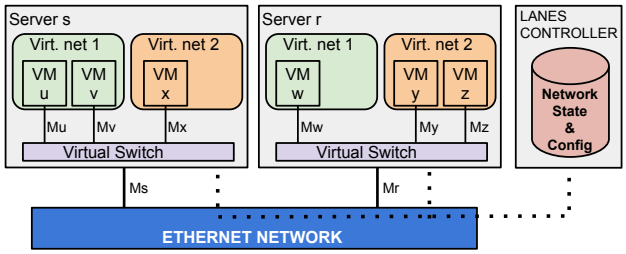


Fig. 1: LANES perspective of an example deployment in a multi-tenant datacenter.

TABLE I: An example flow identifier table for servers in Fig. 1.

OUTBOUND KEY				INBOUND KEY		
SRC MAC	DST MAC	SRC IP	DST IP	FLOW	SRC MAC	DST MAC
M_u	M_w	A_u	A_w	F_{uw}	M_s	M_r
M_w	M_u	A_w	A_u	F_{wu}	M_r	M_s
M_x	M_y	A_x	A_y	F_{xy}	M_s	M_r
M_y	M_x	A_y	A_x	F_{yx}	M_r	M_s

B. Packet forwarding

LANES intercepts all packets that VMs send to perform address virtualization. If a packet matches an entry in the flow identifier table, LANES forwards as indicated by its flow identifier. We now describe how LANES creates flow identifiers for packets with no matches.

LANES requires information about the infrastructure and hosted virtual networks to identify which VMs can communicate and configure flow identifiers accordingly. LANES requires a mapping of VMs to their virtual networks and a mapping of VMs to the infrastructure server where they run. These mappings can be pregenerated according to virtual network configurations or generated on demand as VMs are instantiated, shutdown, or migrated (e.g., when scaling to variable workloads).

LANES also requires a database of which IP addresses are configured in each VM. This database can be pregenerated according to virtual network configurations, or inferred on demand sending ARP packets to all VMs in the tenants's virtual network. Remember that LANES creates one flow identifier for each pair of IPs communicating. To avoid VMs creating arbitrary flow identifiers, the infrastructure provider can limit the number of IPs that a VM can use, or charge for each IP associated with a VM.

In cases where there are no mappings for a packet, LANES determines the packet type, source, and destination. LANES

[2] Inbound and outbound packets can be identified by their input ports, or by checking if the destination MAC address is the server's MAC address.

confirms that source addresses (layers 2 and 3) belong to the sending VM. LANES also checks if source and destination are attached to the same virtual network or if the destination is in an external network (e.g., the Internet). If any of the checks fail, e.g., when VMs are sending packets to non-existing destinations, LANES maps the packet to a special F_{drop} flow identifier. LANES discards all packets that match to F_{drop}. If the packet is to be forwarded, LANES creates a mapping according to the packet's type, source, and destination, as we describe next.

Traffic within a physical server. Authorized traffic between VMs located on the same server is forwarded *without modification* and no packet rewriting is performed. When the first packet of a flow between a pair of IPs belonging to local VMs is intercepted at a server, LANES installs a mapping to a special flow identifier F_{local}.

Traffic between physical servers. When LANES identifies that a packet's destination VM runs on a different server, LANES allocates any unused flow identifier F_{uw} (between the pair of communicating servers) to the pair of communicating IPs. Packets mapping to this flow identifier are rewritten (Sec. II-A) and forwarded to the physical network. LANES also configures F_{uw} at the destination server's virtual switch. Later, when packets reach the destination server, they are rewritten back to their original form and forwarded to the destination VM.

ARP queries. Tenants may use identical, conflicting IP addresses in their VMs. Left unchecked, an ARP query could get multiple answers. LANES not only contains ARP packets to their VM's virtual networks, it answers ARP requests on behalf of VMs to reduce the number of broadcast packets.

IP broadcast traffic. Broadcasts are expensive in large datacenters and can have significant negative performance impact. IP broadcast packets must also be contained to their virtual networks. Consider VM u running on server s sends a broadcast packet $P = [M_u M_\star \mid A_u A_\star]$, where $M_\star$ and $A_\star$ denote broadcast MAC and IP addresses ($A_\star$ has multiple possible values in virtual networks with multiple IP subnets). LANES forwards packet P, unmodified, to any other VM in u's virtual network that is running on server s (as for unicast traffic within a physical server), then uses two mechanisms to transmit broadcast packets between servers.

If the datacenter Ethernet network supports layer-2 multicast, LANES can be configured to create multicast groups for each virtual network. LANES generates a flow identifier $F_{u\star}$ for packet P as described for unicast packets in Sec. II-A, except LANES (i) looks up the multicast group of u's virtual network, denoted M'_u, instead of the destination server's MAC address, and (ii) installs mappings for $F_{u\star}$ on all servers that run VMs attached to v's virtual network. As an example, LANES rewrites P into $P' = [M_s M'_u \mid F_{u\star}]$. This solution requires one entry in switch MAC forwarding tables for each virtual network's multicast group.

If the datacenter does not support multicast or if Ethernet switch MAC forwarding tables cannot handle both server MAC addresses and multicast addresses simultaneously, LANES emulates broadcast using unicast packets. Again, LANES generates a flow identifier $F_{u\star}$ for packet P as described for unicast packets in Sec. II-A, except LANES (i) looks up the MAC addresses of all servers running VMs attached to u's virtual network, and (ii) installs mappings for $F_{u\star}$ on all these servers. LANES then configures servers to transmit multiple unicast packets for each broadcast packet. Emulating broadcast requires no additional entries in switch MAC forwarding tables and avoids the overhead of configuring multicast groups; this solution is preferable to multicast for virtual networks that span a small number of servers.

TABLE II: An example flow identifier table for broadcast and external packets for servers in Fig. 1.

OUTBOUND KEY				INBOUND KEY		
SRC MAC	DST MAC	SRC IP	DST IP	FLOW ID	SRC MAC	DST MAC
Broadcast from VM u:						
M_u	$M_\star$	A_u	$A_\star$	$F_{u\star}$	M_s	M'_s
External connections from VM u:						
M_u	M_χ	A_u	—	$F_{u\chi}$	M_s	M_χ
M_χ	M_u	—	A_u	$F_{\chi u}$	M_χ	M_s

Servers that receive broadcast packets from VMs in other servers use the mappings for $F_{u\star}$ to rewrite P' into P, then forward P to all local VMs attached to u's virtual network. We show an example flow identifier mapping for broadcast packets in Tab. II.

If LANES can map VMs to their IP subnets, then LANES can reduce network load by creating multicast groups for each IP subnet (at the cost of additional switch MAC forwarding table entries) or send unicast packets only to servers running VMs in that IP subnet. IP subnets could be specified in virtual network configurations or inferred from DHCP messages in virtual networks configured using DHCP.

Traffic to external networks. LANES allows VMs to communicate with the Internet or with VMs in other virtual networks by placing layer-3 routers at virtual network boundaries. These border routers serve as gateways. Packets to and from external networks are identified by having the border router's MAC address, denoted M_χ.

Consider VM u in server s sends packet $P = [M_u M_\chi \mid A_u A_\chi]$ to an external host A_χ. LANES generates an *external* flow identifier $F_{u\chi}$ for packets between VM u and A_χ. To avoid generating one flow identifier whenever a VM connects to a different external IP address, we note that LANES needs to virtualize the VM's MAC and IP addresses, but not the gateway's MAC address or the destination's IP address. LANES builds external flow identifiers $F_{u\chi}$ with 32 bits to virtualize VM addresses. External flow identifiers overwrite the source IP address of outbound packets and the destination IP address of inbound packets. LANES keeps the external IP address A_χ untouched when rewriting inbound and outbound packets. As an example, LANES rewrites packet P as $P' = [M_s M_\chi \mid F_{u\chi} A_\chi]$. Tab. II shows example external flow mappings.

LANES allows tenants with multiple virtual networks to attach VMs to more than one virtual network and route packets between their own virtual networks. As would be expected in any network, unless a VM's source IP address A_u is globally-reachable and routed to the infrastructure provider's datacenter, the tenant must provide the means for an address to be translated (using NAT) or for the packet to be tunnelled (e.g., in a VPN). In such cases, the NAT/VPN server would be part of the tenant's network and would be reachable through LANES.

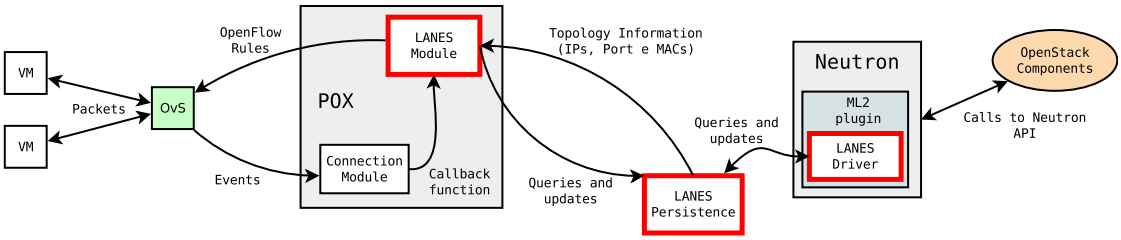


Fig. 2: Applications that compose LANES and interaction forms between them and the IaaS platform.

III. Implementation

We implemented a LANES prototype on top of OpenStack, using the POX SDN controller. OpenStack is one of the most well known open IaaS platforms; it provides applications and APIs that virtualize datacenter infrastructures, covering tasks such as server allocation, network definition, access control, firewalls, and high availability. OpenStack's *Neutron* acts as a network abstraction layer, providing APIs for tenants to express their virtual network topologies, which can then become accessible to other components.

We implemented LANES in three major modules as shown in Fig. 2. LANES's OpenStack driver implements the functions necessary to connect to Neutron and obtain virtual network topologies. LANES's network controller executes on top of the POX SDN controller and uses OpenFlow to control Open vSwitch instances running on each server in the infrastructure. LANES's persistence module keeps all topology information in a distributed database for efficient access and robustness.

Network changes such as VM instantiation or migration are sent to Neutron. Neutron, in turn, propagates changes to LANES, which can reconfigure Open vSwitch instances as necessary to guarantee correct packet delivery. When a virtualization server first boots, its Open vSwitch instance contacts the LANES controller, which configures that instance and adds it to its database. Open vSwitch instances inform the LANES network controller of any changes to its ports (e.g., link up and link down events). This allows LANES to obtain all information it needs to build flow identifiers: map VMs to virtual networks, map VMs to servers, and map servers to their MAC addresses.

IV. System Evaluation

This section describes the environment built to evaluate LANES's capabilities and performance. We show that LANES achieves forwarding performance equivalent to existing tools with negligible memory overhead while providing isolation and reducing MAC address table pressure on switches.

A. Testing environment

A physical infrastructure corresponding to part of the infrastructure of an IaaS provider was built to validate LANES's operation. Figure 3 presents the testing environment topology. It shows three virtualization servers connected to two switches. One switch is used for OpenStack control communications, to access the datacenter network, to communicate with the POX controller, and to exchange traffic with the Internet. The second switch is exclusively used for traffic between virtual

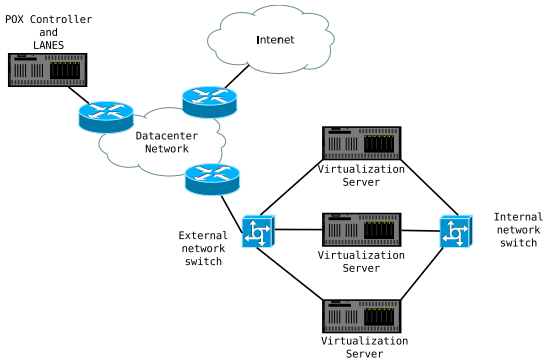


Fig. 3: Physical topology of the testing environment.

machines, corresponding to the most probable configuration of an IaaS provider. IaaS providers commonly have very big infrastructures, where servers and controllers are spread across the network. It is important to note that both switches are commodity learning Ethernet switches, without any special configuration.

We chose two switches to facilitate the understanding and verification of the correct isolation behavior between the physical and virtualized environment. The virtualization servers are three Intel Xeon E5440. The POX controller runs on an Intel Xeon E3-1240. All machines have 1 Gbps network interfaces and are connected to 1 Gbps switches. The virtual machines used in all the tests were configured with 1 processor, 1 Gbyte of RAM, 30 Gbytes of storage, and Ubuntu Linux 13.10.

On top of the physical structure, we deployed multiple virtual networks, from different tenants. Each virtual network contains multiple VMs and may span multiple servers. We considered three different software stacks for the network configuration:

1) LANES with POX module;
2) L2 switch from POX, which is offered as a reference, indicated as POX+L2;
3) OvS switch. It only forwards packets in Ethernet. It had no configured controller nor isolation between virtual networks.

In the following we evaluate isolation properties, flow establishment latency, forwarding bandwidth (in packets per second), and load overhead.

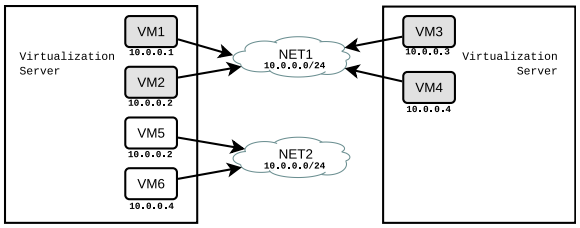


Fig. 4: Network topology during isolation tests.

TABLE III: Network isolation tests with LANES for the configuration in Fig. 4: only VMs in the same network reply to echo requests.

Sent by \ Received by	VM_1	VM_2	VM_3	VM_4	VM_5	VM_6
VM_1	–	√	√	√		
VM_2	√	–	√	√		
VM_3	√	√	–	√		
VM_4	√	√	√	–		
VM_5					–	√
VM_6					√	–

B. Isolation between virtual networks

The test for validating the isolation between virtual networks was carried out by sending ICMP packets from multiple virtual machines of different tenants using the same IP address block (10.0.0.0/24). The topology presented in Fig. 4 shows the evaluation environment, which contains a six virtual machines distributed between two tenant virtual networks. Since the networks are independent, it is expected that even using the same IP, a tenant's packet does not reach VMs in the other tenant's network.

The test consists of each VM issuing ICMP ping (echo) requests to the network broadcast address. In each column of Tab. III, we mark the virtual machines that replied to the ICMP echo request packet sent by the VM in each row. As we can see, LANES allowed the requests to reach only the machines that belong to the tenant's virtual network.

While LANES provides isolation between tenant virtual networks, POX+L2 and OvS do not. Tab. IV shows results for the same test under POX+L2 and OvS network stacks. Both mechanisms do not provide isolation and behave similarly, with all VMs from both networks receiving and replying to all ICMP echo requests.

To demonstrate the importance of isolation, we performed a throughput test. We configured a TCP flow between VM_1 and VM_3, and configured VM_6 to flood its own network with broadcast packets. That might be the behavior of a misconfigured application in its own network, or might even be an intentional malicious DoS attach on the other tenant's network. Tab. V presents the observed throughput for the tested tenant's TCP flow under each configuration. Since LANES limits the flood to that tenant's network, within one server in the infrastructure, LANES can provide better performance to the other tenant than the other two configurations.

C. Latency

The objective of latency tests is to measure the time before the start of communication between two virtual machines inside the infrastructure. During this phase many actions occur, like MAC address discovery, installation of flow rules into the OvS switches, packet rewriting (if necessary), and packet forwarding.

TABLE IV: Network isolation tests with both POX+L2 and OsS stacks for the configuration in Fig. 4: all VMs in the both network reply to echo requests.

Sent by \ Received by	VM_1	VM_2	VM_3	VM_4	VM_5	VM_6
VM_1	–	√	√	√	√	√
VM_2	√	–	√	√	√	√
VM_3	√	√	–	√	√	√
VM_4	√	√	√	–	√	√
VM_5	√	√	√	√	–	√
VM_6	√	√	√	√	√	–

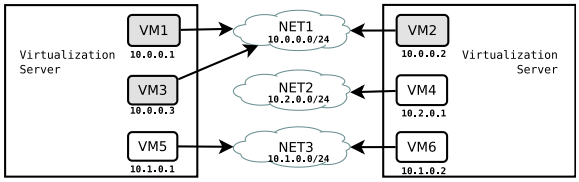


Fig. 5: Virtual network topologies used to evaluate MAC address resolution. All logical networks were implemented over the physical network in Fig. 3.

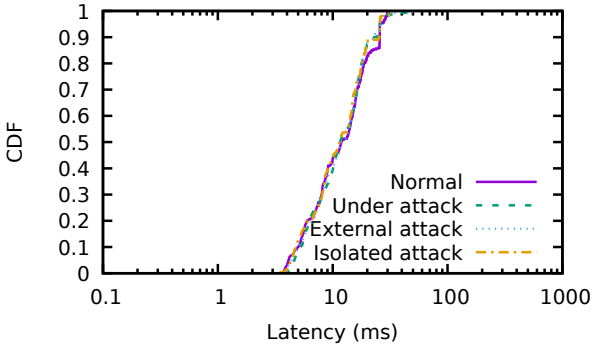
All stages prior to the beginning of communication were evaluated separately to provide a better understanding of the system's behavior. We measured the time required for MAC resolution by the ARP protocol, the time required for installing the flow rules and the time required for forwarding the packets after the flow was established. We also measured the forwarding latency of broadcast packets over unicast packets, since this type of transmission requires software processing by LANES.

MAC address resolution. Because LANES intercepts and answers all ARP packets without contacting the destination VM, this communication stage was measured separately. We measured the time interval between a VM sending an *ARP Request* and receiving an *ARP Reply*. All measurements were done capturing traffic at the VM's (virtual) network interface. The following MAC resolution tests were done using the topology presented in Fig. 5:

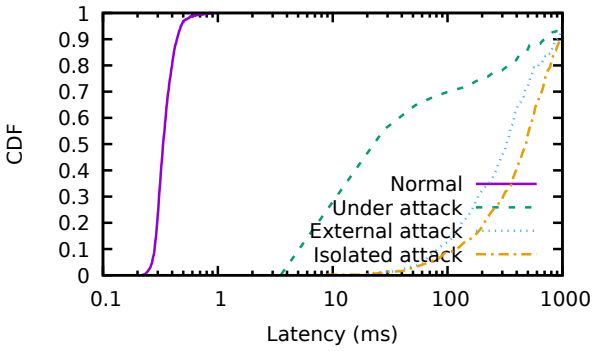
1) Normal operation: VM_1 requests the MAC address of VM_2 on the same virtual network;
2) Attack on the local network: VM_1 requests the MAC address of VM_2 while VM_3 is flooding the same local network NET_1;
3) Attack on remote network: VM_1 requests MAC address of VM_2 while VM_5 is flooding network NET_3;

TABLE V: TCP throughput between VM_1 and VM_3 in Fig. 4 while VM_6 floods its virtual network with broadcast packets.

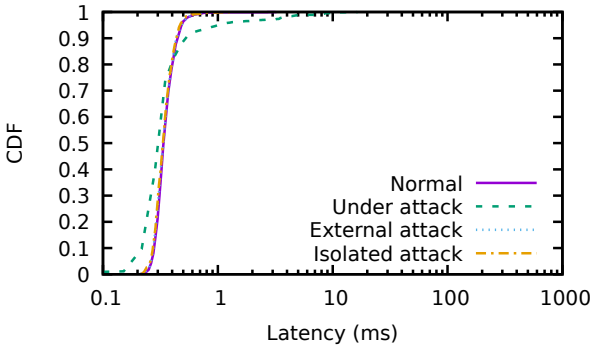
Configuration	Average Throughput	Std. Dev.
LANES	825 Mbps	2.1
POX+Switch L2	170 Mbps	5.3
OvS	295 Mbps	3.2



(a) LANES



(b) POX+L2



(c) OvS

Fig. 6: ARP resolution latency.

4) Attack on isolated network: VM_1 requests MAC address of VM_2 while VM_4 is flooding network NET_2. The difference between tests 3 and 4 is that in test 4 the attack does not generate inter-server traffic.

Figure 6 shows the results of ARP resolution using LANES, POX+L2, and OvS. Each figure contains four lines presenting results for each test.

As expected and confirmed by Fig. 6, LANES had similar response times in every type of test. This happens because LANES intercepts the ARP traffic, generates the responses internally, and prevents performance degradation due to the flooding attacks. Compared to other scenarios, response times are higher in the normal case, since the controller has to participate in all ARP processing.

In the situation where the POX controller is used with the L2 switch application, the system performance degrades

Fig. 7: Topology used to evaluate flow configuration time, packet forwarding latency, and broadcast latency.

TABLE VI: Packet forwarding latency for established flows between VM_1 and the other VMs in Fig. 7 (in ms).

Dest.	Conf.	Avg.	Max	Min	Std.
VM_2	LANES	0.35	1.19	0.26	0.078
VM_2	POX+L2	0.36	0.99	0.25	0.079
VM_2	OvS	0.37	0.95	0.27	0.080
VM_3	LANES	0.32	0.89	0.21	0.070
VM_3	POX+L2	0.32	0.74	0.21	0.063
VM_3	OvS	0.31	0.70	0.21	0.064

significantly under flood attacks, because all traffic is reaching all ports, and that must be processed by the L2 switch module. For that reason, results in Fig. 6 show that POX+L2 has the worst performance for the attacks tested.

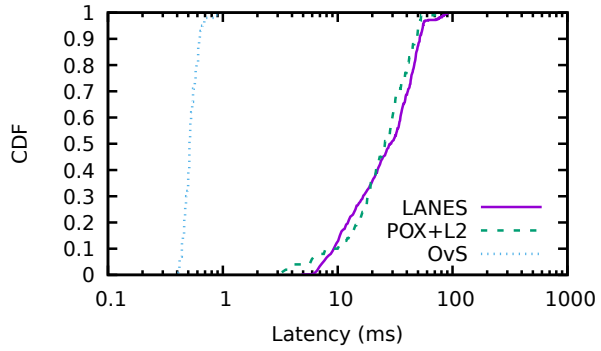
Open vSwitch has the best results in all cases. That is possible mainly because it executes within the operating system kernel and also because no type of validation is done. On the other hand, all switches and network connections are overloaded during the attacks, because it unconditionally forwards every packet that is generated during the flood.

Flow configuration latency. We also evaluate the time it takes LANES to compute flow IDs and configure flow forwading rules. As LANES rewriting rules require some computation by the controller and are installed in both the source and destination virtual switches, we expect some performance loss relative to POX+L2. We evaluate flow configuration latency on the topology shown in Fig. 7.

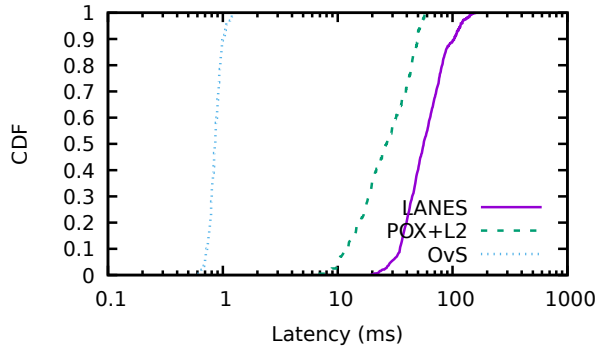
Figure 8 shows the results. In all situations, the use of a controller that installs forwarding rules reactively causes a longer delay when compared to an OvS switch without a controller. Figure 8(b) shows that configuring rules for inter-server communication has higher latency, as it requires installing rules in two OvS instances. LANES has performance similar to that of POX+L2 and we note that the latency overhead is reasonable, since flow configuration is incurred only once for each pair of communicating IP addresses (during the VMs initialization process).

Packet forwarding latency. We evaluate packet forwarding latency for established flows using the topology in Fig. 7. We measure the latency between VM_1 and VM_2 as well as VM_3.

Tab. VI shows average, maximum, minimum, and the standard deviation of packet forwarding latency for all flows and different configurations. We observe LANES forwarding overhead is minimal compared to other networking stacks, with the benefit of isolation between networks. This is relevant as forwarding overhead is incurred on each packet, while ARP resolution and flow configuration happen only once for each pair of communicating IP addresses.



(a) Same server



(b) Different servers

Fig. 8: Latency for installing OpenFlow rules on switches

Broadcast latency. We also evaluate LANES's performance for broadcast traffic emulated using unicast packets in the network topology in Fig. 7. The latency evaluation was performed sending broadcast ping packets from VM_1.

Fig. 9 shows broadcast latency for different tests. One may observe the delays added by including an SDN controller by comparing the POX+L2 and OvS lines. The reader can also observe the broadcast of LANES's software emulation of broadcast packets by comparing the LANES and OvS lines. Our current implementation in LANES emulates broadcasts using unicast packets from userspace, which achieves an average performance of $0.563\ ms$, while the SDN configuration using POX+L2 is $0.035\ ms$. We note LANES's performance could be improved by implementing the emulation of broadcast over unicast in kernel-space or by performing broadcast using multicast. As the OvS without controller does not check anything, its results were significantly better than the other two, obtaining an average response $0.0056\ ms$.

Although slower to emulate, the emulation of broadcast using unicast is worth considering in large environments, where each tenant usually has few VMs relative to the size of the datacenter. In these scenarios, emulating broadcast using unicast packets prevents the tenant from flooding the entire datacenter network. We also note that there may be only a few reasons for broadcast packets left as LANES handles ARP separately.

TABLE VII: Available bandwidth between VM_1 and the other VMs in Fig. 7 (in Gbps).

Dest.	Conf.	Avg.	Max	Min	Std.
VM_2	LANES	0.94	0.95	0.63	0.02
VM_2	POX+L2	0.93	0.94	0.74	0.02
VM_2	OvS	0.93	0.94	0.93	0.00
VM_3	LANES	0.94	1.05	0.84	0.01
VM_3	POX+L2	0.94	1.14	0.76	0.03
VM_3	OvS	0.94	0.95	0.92	0.00

D. Available bandwidth

We measured the maximum achievable TCP throughput using the same topology as in Fig. 7. We use `iperf` to generate synthetic traffic and measure bandwidth between VM_1 and the other two virtual machines, VM_2 and VM_3. The TCP connection was tested for 10 hours.

Tab. VII shows achieved throughput in each scenario. The results demonstrate that the use of an SDN controller does not influence the results and that LANES's (negligible) packet forwarding latency has negligible impact on throughput. Observe that for connections between VMs in the same virtualization server, the maximum throughput is limited only in software by OvS. Thus, the results in these cases can reach values higher than the maximum expected throughput for connections (1 Gbps).

E. Scalability evaluation

The last set of tests evaluates the system capacity, in particular, how the POX controller deals with the demand of new flow rules. The topology was composed of four virtual machines distributed over two virtualization servers and connected to the same virtual network (as in NET_1 in Fig. 4).

We configure VM_4 to generate an increasing rate of TCP connections over consecutive 120-second rounds. We show the connection generation rate and round start times in the vertical bars in the upper graph in Fig. 10. TCP flows are established with any other VM at random. LANES will then configure rules for all connections so VMs can exchange traffic. We measure the CPU utilization of the OvS process in VM_4 as it handles the highest workload and show measurements using green triangles in the upper graph of Fig. 10. The middle graph shows aggregate TCP bandwidth. The bottom graph shows ping latency, which represents flow configuration delay. The x-axis of all graphs are aligned and cover the experiment duration.

The measurement results for LANES, implemented on the POX controller, are shown in Fig. 10. They show that the developed system begins to suffer performance degradation when the rate of TCP connections is about 200 connections per second. Although it is only capable of handling well a low amount of new connections, we must remember that LANES performs several checks before deciding which flows need to be configured. The use of a production-grade controller, like ONOS, would certainly improve performance in this case.

Since LANES depends on the topology implemented in OpenStack and also needs to access the database to query it,

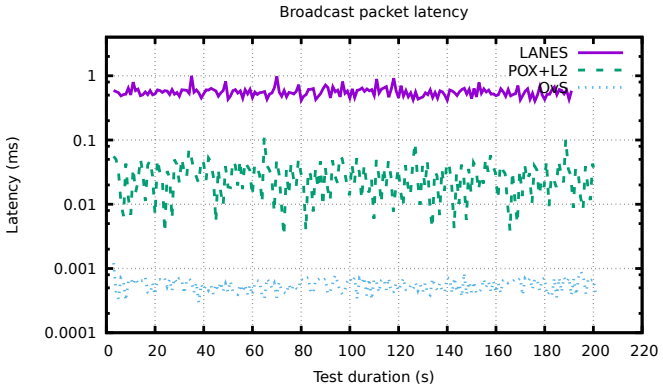


Fig. 9: Broadcast latency.

TABLE VIII: Comparison of SDN-based datacenter network virtualization solutions ('yes' is preferable).

PROPERTY	LANES	NVP	NetLord	Diverter
No encapsulation	Yes	No [12]	No [3]	Yes [13]
No header modification	No	Yes	Yes	No
IP virtualization	Yes	Yes	Yes	No
Non-IP traffic	No	Yes	Yes	No
On demand config	Yes	No	Yes	Yes
No fragmentation	Yes	Yes	No	Yes
External flows	Yes	Yes	No	Yes

LANES has an overhead higher than other network configuration stacks to handle OpenFlow messages. As LANES has not been implemented with high performance as its main objective, given POX has been implemented in Python, we believe that LANES can be improved by adjusting the used components or rewriting the functions with higher CPU usage.

V. RELATED WORK

GRE [9], and MPLS [10] require extra header fields to encapsulate packets and to provide traffic isolation. Moreover, these approaches have a limited network segment space. For instance, VLAN fields have 12 bits and allow only 4096 VLAN tags, severely limiting scalability. Another possible isolation solution through encapsulation is Q-in-Q [11], which also makes use of VLAN tags, but it uses two tags instead of one. This solution allows a much larger amount of isolated networks to run in parallel, but may still be insufficient for large datacenters. In addition, as with VLANs, the number of virtual machines in the datacenter can be so great that the switches will be unable to store all network MAC addresses in their forwarding tables. LANES does not require additional header fields and provides significantly better scalability.

NetLord [3] adopted a solution similar to ours to reduce the size of routing tables in switches inside a datacenter. NetLord encapsulates packets leaving virtual machines, creating a new packet header, addressing them to the final server in which the destination VM is located. By encapsulating the packet during transmission by the network, NetLord prevents the MAC addresses of virtual machines to be registered by switches on the way. Thus, only server addresses are stored in switch forwarding tables. Traffic encapsulation was necessary to enable one of the main features of NetLord, which is the use

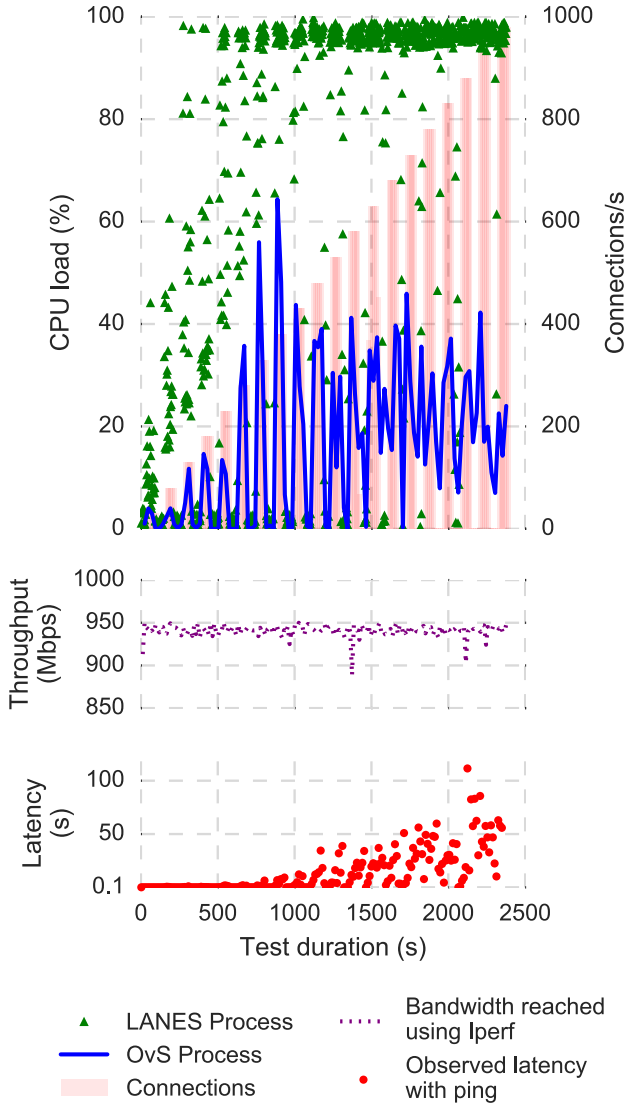


Fig. 10: Load test results for LANES.

of multiple simultaneous routes between devices. SPAIN [14] is an application by the same group that takes advantage of multiple paths to route traffic more efficiently and improve bandwidth. The main differences between LANES and NetLord is that NetLord uses encapsulation, which increases packet sizes and may cause fragmentation, and it requires changes to the virtualization layer. NetLord also does not allow direct communication between different virtual networks.

Diverter [13] is an application developed by HP Labs, which has similar objectives to LANES, such as allowing virtual networks to share the physical infrastructure while ensuring isolation. An important difference between Diverter and LANES is that Diverter assigns IP addresses to VMs; this simplifies isolation as it prevents IP address collision between different VMs. LANES allows tenants to configure arbitrary IP addresses on VMs.

Some of the authors of the initial works on SDN [15] presented NVP [12], a system to virtualize datacenter networks. The tool uses the SDN paradigm and uses network

topology information created by each customer to build a traffic isolation solution between participants. Unlike LANES, which works reactively, NVP precomputes all flow rules based on virtual network topology configurations. The controller only communicates with the switches when the topology changes, which requires change to the configured rules. Flow rule computation costs are significant, as it is necessary to cover all possible communication possibilities, ignoring the fact that VMs rarely establish all possible flows [16]. The authors report that in a network with 3,000 virtualization servers and more than 60,000 ports, flow rule computation took up to one hour to complete. In NVP, isolation between clients on the physical network is achieved using encapsulation. LANES provides on-demand configuration of flow forwarding rules and uses only packet rewriting. We summarize the trade-offs between different SDN-based datacenter network virtualization solutions in Tab. VIII.

Other previous works have proposed new traffic control and resource (bandwidth) allocation solutions for datacenter networks [17]–[20]. Those solutions allow an IaaS provider to allocate resources to tenant virtual networks and guarantee performance bounds. LANES is orthogonal to these solutions, providing the network isolation they require, and it can be combined with traffic control and resource allocation solutions to provide a richer IaaS environment.

VI. Conclusions

In this work we presented LANES, a system designed to provision and isolate virtual networks in datacenter environments. LANES rewrites packet headers to virtualize addresses, providing flexibility to hosted VMs and preventing VMs from directly impacting the physical network. LANES requires no modification to hosted VMs, puts no restrictions on VM IP addresses, does not incur encapsulation overhead, and scales to millions of VMs and thousands of servers in a single Ethernet segment. LANES also does not require advanced features and works on top of commodity Ethernet switches. LANES provides these benefits at the cost of memory utilization on infrastructure servers (hundreds of KBs), increased latencies during flow setup (hundreds of milliseconds, once per IP pair), and compute overhead to rewrite packet headers.

We integrate LANES with OpenStack, a widely used IaaS platform. LANES implements an SDN controller responsible for orchestrating the use of network resources. LANES uses Open vSwitch and standard OpenFlow rules to rewrite packets at infrastructure servers and virtualize network addresses.

We evaluated our implementation of LANES in a physical testbed. The results show that LANES effectively isolates traffic between different virtual networks. We also quantified system performance under heavy workloads. We showed that LANES can be effective in protecting the network from DoS attacks within the datacenter network. Finally, the results show that LANES achieves performance similar to that of simpler solutions after flow rules are installed.

In the future, we plan to evaluate the advantages of proactively generating rules. Although such behavior has a high flow computation cost, it has been advocated by SDN creators as preferable whenever possible.

Acknowledgments

This work was funded by FAPEMIG, CNPq, CAPES, and by projects InWeb (MCT/CNPq 573871/2008-6), MASWeb (FAPEMIG-PRONEX APQ-01400-14), and EUBra-BIGSEA (H2020-EU.2.1.1 690116, Brazil/MCTI/RNP GA-000650/04).

References

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *ACM SIGCOMM CCR*, vol. 39, no. 1, pp. 68–73, 2008.

[2] M. Yu, J. Rexford, X. Sun, S. Rao, and N. Feamster, "A survey of virtual lan usage in campus networks," *IEEE Comm. Mag.*, vol. 49, no. 7, pp. 98–103, 2011.

[3] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, "NetLord: A Scalable Multi-tenant Network Architecture for Virtualized Datacenters," in *Proc. ACM SIGCOMM*, 2011.

[4] T. Sridhar, L. Kreeger, D. Dutt, C. Wright, M. Bursell, M. Mahalingam, P. Agarwal, and K. Duda, "VxLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," IETF, Tech. Rep., 2014.

[5] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, "Virtualizing the Network Forwarding Plane," in *Proc. Workshop Programmable Routers for Extensible Services of Tomorrow*, 2010.

[6] OpenStack, *OpenStack Networking Administration Guide*, OpenStack Foundation, 2013.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[8] T. Narten, M. Karir, and I. Foo, "Address resolution problems in large data center networks," RFC 6820, IETF, 2013.

[9] D. D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784, IETF, 2000.

[10] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, IETF, 2001.

[11] T. Jeffree, "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks—Amendment 4: Provider Bridges," 2006.

[12] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, N. Gude, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang, "Network Virtualization in Multi-tenant Datacenters," in *Proc. USENIX NSDI*, 2014.

[13] A. Edwards, A. Fischer, and A. Lain, "Diverter: A New Approach to Networking Within Virtualized Infrastructures," in *Proc. ACM Workshop Research on Enterprise Networking*, 2009.

[14] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS Data-center Ethernet for Multipathing over Arbitrary Topologies," in *Proc. USENIX NSDI*, 2010.

[15] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking Control of the Enterprise," *ACM SIGCOMM CCR*, vol. 37, no. 4, pp. 1–12, 2007.

[16] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The Nature of Data Center Traffic: Measurements & Analysis," in *Proc. ACM IMC*, 2009.

[17] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks," in *Proc. Usenix Workshop on I/O Virtualization*, 2011.

[18] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying NOX to the Datacenter," in *Proc. ACM HotNets*, 2009.

[19] B. Heller, D. Erickson, N. McKeown, R. Griffith, I. Ganichev, S. Whyte, K. Zarifis, D. Moon, S. Shenker, and S. Stuart, "Ripcord: a modular platform for data center networking," *ACM SIGCOMM CCR*, vol. 40, no. 4, pp. 457–458, 2010.

[20] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "FairCloud: Sharing the Network in Cloud Computing," in *Proc. ACM SIGCOMM*, 2012.

FLOWer – Device Benchmarking Beyond 100 Gbit/s

Paul Emmerich, Sebastian Gallenmüller, and Georg Carle
Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich
{*emmericp*|*gallenmu*|*carle*}*@net.in.tum.de*

Abstract

The growth of bandwidth in computer networks fuels the constant adoption of measurement equipment and methodologies. Networking equipment offers processing capacity in the Terabit/s range nowadays. However, test equipment in academia lags behind. We propose FLOWer, a novel and cost effective approach capable of testing such high-speed devices. FLOWer combines an inexpensive software packet generator with an OpenFlow-enabled switch to amplify the bandwidth while sustaining the flexibility of the software solution. By utilizing OpenFlow, FLOWer is able to provide the required bandwidths the software solution cannot generate on its own. We demonstrate a proof-of-concept with example measurements at bandwidths of multiple 100 Gbit/s.

1. Introduction

With data centers deploying 10 GbE as standard equipment and with the increasing availability of succeeding standards like 40 GbE and 100 GbE the demand for bandwidth in network devices is rising. At the same time, technologies like OpenFlow and network function virtualization introduce new possibilities to configure these devices. However, this increased flexibility comes with a price, as there is a tradeoff between performance and flexibility. This situation creates the need for new benchmarking and testing methodologies to enable an accurate assessment of these devices.

We propose a new way to efficiently test devices with bandwidths in the Terabit/s range with little effort. To achieve this FLOWer uses a combination of a packet generator with an OpenFlow switch. This combined system offers the flexibility of a software packet generator and at the same time the performance of a hardware solution through the OpenFlow-enabled switch. The software-generated packets are fed into an OpenFlow-enabled switch which can multiply them manifold via OpenFlow and generate high bandwidths in a simple and elegant way. OpenFlow capabilities also allow modification or the measurement of the traffic.

Our evaluation is based on modern 10 Gbit/s OpenFlow hardware. Relevant features and limitations of these devices are discussed further in Section 3. The most basic test setup uses the capabilities of FLOWer to perform self-tests on OpenFlow-enabled devices by connecting the FLOWer switch with itself. We show this setup with example measurements in Section 4. More complex test setups can be achieved by connecting the FLOWer switch to other devices. A sample setup is discussed in Section 5. All results and code used for the experiments presented here are publicly available [5], cf. Section 6.

2. Related Work

Packet generators face a trade-off between flexibility and performance. Software packet generators are typically slow and unreliable [2]. Hardware packet generators offer high precision, speed, and number of ports [21]. However, they lack the flexibility of modern software packet generators that can be configured with scripts [7]. Specialized hardware is always expensive compared to the commodity hardware required for software tools (cf. Section 3.3). The software packet generator MoonGen [7] solves this problem to some extent by using hardware features found on commodity server network interface cards (NICs) to provide high precision. However, the speed and number of ports still lags behind commercial hardware offerings as it is restricted to server hardware.

Professional hardware packet generators offer a large number of ports and even multiple 100 GbE ports [22]. The need for higher speeds is apparent in the literature. E.g., Rotsos et al. present OFLOPS, a framework to test OpenFlow switches with a NetFPGA [20]. Their original framework was limited to 100 Mbit/s on GbE ports and later extended to 20 Gbit/s [19] with the OSNT packet generator [1]. However, the devices they are testing offer speeds beyond 100 Gbit/s. FLOWer is not an alternative to existing packet generators, but an addition: it can be combined with a framework like OFLOPS to solve this discrepancy.

A technique similar to FLOWer was used by Mahadevan et al. in 2009 [11]. They wired a switch in a way that all ports were connected back to the switch itself. Broadcast traffic sent over this configuration loads the switch without requiring a high performance traffic generator. We use a similar wiring approach for self testing the FLOWer switch (cf. Section 4) However, our approach is more precise as it uses OpenFlow features to shape unicast traffic to our specific requirements and to measure throughput.

ISBN 978-3-901882-84-5, IFIP Networking 2016

Kuźniar et al. discuss characteristics of OpenFlow flow table implementations on different switches [10]. The results like the flow table size and costs of modification operations are important for us to select a suitable OpenFlow switch.

3. Test Hardware and Software

In the following the hard- and software is presented which we used for our proof of concept. However, FLOWer does not depend on these specific devices. It is rather a methodology and can be transferred to other OpenFlow-enabled hardware and other packet generators.

3.1. OpenFlow Switches

OpenFlow specifies programmable switches: they can be configured to match packet headers based on *flows*. These flows are also referred to as rules in this paper. The packets matching against these flows can be modified via modification actions and can then be sent out on one or more ports. Traffic can be accounted through statistics associated with flows. [13]

We use an Edge-Core Networks AS5712-54X 10 GbE switch which is based on Broadcom BCM56854 Trident II switch ASICs with 48 10 GbE ports and 6 40 GbE ports [4] for our proof of concept. This switch is a design approved by the Open Compute Project, several switches with similar designs are available on the market [12][1]. The standardized design allows for multiple choices of operating systems while keeping the hardware costs low. We selected the PicOS operating system as it features a mature implementation of OpenFlow 1.4 [18].

This hard- and software allows for benchmarking speeds of up to 720 Gbit/s. However, we did not have 40 GbE cables in stock, so we were restricted to 480 Gbit/s here. Even this speed and port density is beyond the capabilities of packet generators that are usually available in the academic field.

3.2. MoonGen Packet Generator

We use our packet generator MoonGen [6], as it is a highly flexible software packet generator: it crafts all packets in real-time with user-provided Lua scripts. It also features latency measurements with sub-microsecond precision by using timestamping features on commodity NICs. [7]

High-speed packet generation is not required for FLOWer, but it simplifies some test setups if the packet generator can keep up with the fastest port on the switch. We use MoonGen with two 10 GbE interfaces, which it is able to saturate on a low-end Xeon E3-1230 v2 with a dual port Intel X520-T2 NIC. MoonGen is capable of generating bandwidth of up to 180 Mpps at 120 Gbit/s on commodity hardware, so it is scalable to 100 GbE switches. Readers interested in more details about MoonGen are referred to our full evaluation. [7]

1. This switch is listed as "Accton AS5712-54X" in the specification; Edge-Core Networks is a subsidiary of Accton.

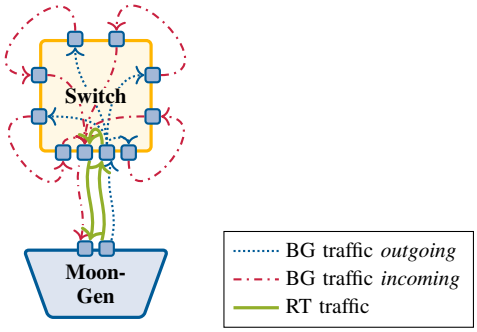


Figure 1. Self-flooding test setup

3.3. Cost Effectiveness

The total cost of the test setup (without the device under test (DuT)) to benchmark devices at 710 Gbit/s with this switch was less than € 10 000. A test setups for several Tbit/s using newer 100 GbE switches can be built for less than € 30 000.

These prices are based on quotes from 2015 and are expected to drop even further below prices for hardware packet generators[2] as OpenFlow switches and 40/100 GbE networks become more commonplace.

4. Self-testing SDN Devices

OpenFlow can be used for self-tests of devices. This allows testing an OpenFlow-enabled switch at maximum rate, without requiring any additional hardware beside the device under test, in this case the switch itself, and a software packet generator running on commodity hardware.

We present possible configurations and example measurements here. A switch can be wired and programmed such that it sends traffic to itself. We connected port 47 and 48 to our packet generator and port 1 with port 2, port 3 with port 4, etc.

4.1. Evaluating Quality of Service Features

Switches can define multiple queues per port with different priorities to implement quality of service (QoS). Such hardware features on switches enable modern implementations of QoS in data centers, e.g. by using IEEE 802.1Q [9] service classes mapped to queues on a switch [8]. FLOWer allows us to test hardware features like this under extreme circumstances as this example measurement demonstrates.

4.1.1. Test Setup. We generate two network flows of minimum-sized UDP packets with different destination UDP ports at our packet generator. Figure 1 merely illustrates

2. A search on eBay suggests that a second-hand Spirent TestCenter packet generator, without any software licenses, with a comparable number of ports and speeds costs far more than € 100 000.

wiring and traffic flow not the actual setup. This sketch has a reduced number of ports and does not represent the full duplex transmission.

The first network flow, the *real-time (RT) flow*, is to be prioritized and sent with a constant rate of 1 Gbit/s. This flow is matched by a rule on switch port 48 and sent directly back to the packet generator on switch port 47 via a high-priority queue (cf. the solid line in Figure 1).

The rate of the second network flow, the *background (BG) flow*, is varied in this experiment. The switch is configured to send it out on ports 1 to 46, as depicted with dotted lines in Figure 1. From there it flows back to these ports via the external cabling (cf. the slash dotted line in Figure 1). This amplifies the traffic 46-fold. All incoming traffic from these ports is sent back to the packet generator via a low-priority queue. This tests the behavior of a prioritized network flow under increasing load of unimportant background traffic.

The packet generator then measures the latency of both network flows. Note that the packet generator receives up to 46 copies for each packet it sends in the background which is a potential challenge for timestamping. MoonGen uses sequence numbers for timestamping and defines the latency as the time until the first copy of a packet arrives back at the packet generator. Subsequent packets with the same sequence number are ignored, this is therefore the best-case latency of the background traffic.

Our repository [5] contains the script `selftest/qos.sh` which was used to install the OpenFlow flows on the DuT.

4.1.2. Test Results. Figure 2 shows the latencies of the two network flows with the QoS queue enabled and disabled. It demonstrates that the QoS features work with hundreds of Gbit/s BG traffic.

The deviation of about 700 ns between background and real-time traffic for low rates in both tests is a result of the test setup: the background traffic flows through an additional hop during the amplification step while the real-time traffic is forwarded directly back to the packet generator (cf. Figure 1).

Another result of this test is that the RT traffic is affected by the presence of BG traffic even with QoS enabled. Inspecting the histograms of the RT traffic's latency reveals that it follows a bimodal distribution. Figure 3 shows the latency under a background load of 8 Gbit/s with two clearly visible peaks. As the switch operates in cut through mode, the left peak represents the immediate transfer of a packet. The right peak shows delayed transfer due to ongoing BG traffic transmission, resulting in this bimodal distribution.

Figure 4 on the next page shows the latency distributions for other ratios of RT to BG traffic as cumulative distribution functions (CDFs). The amount of packets that must be queued increases with the BG traffic, i.e., the ratio of the peaks in the distribution changes with the ratio of the traffic. All packets must be queued once the link is saturated, so the QoS features works best when the link is not overloaded.

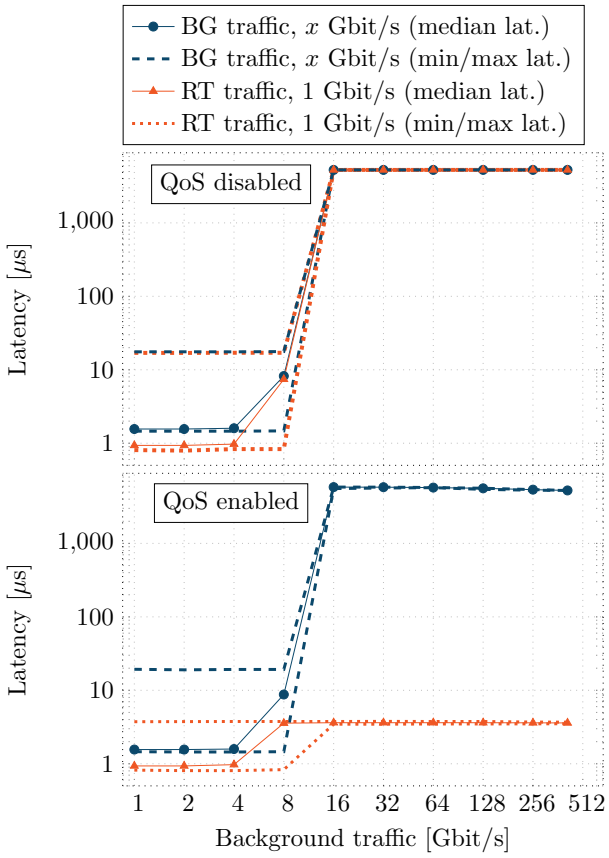


Figure 2. Forwarding latencies with and without QoS

4.2. Forwarding Performance and Latency

Another self-test scenario is forwarding packets in a loop (Figure 5). The packet generator sends on both ports to generate a total bidirectional load of 480 Gbit/s on the switch. All internal connections are realized with OpenFlow flows.

4.2.1. Forwarding Performance. We added OpenFlow rules that match addresses from layer 1 (switch ports) to layer 4 (UDP ports) with modifications of all supported header fields from layer 2 to 4 for all packets to maximize the system load.

The switch achieved line rate in all tested configurations. Some OpenFlow switches perform a fallback to a software implementation for operations not supported in hardware at the expense of performance [20]. PicOS only accepts OpenFlow flows supported in hardware in line rate [18].

4.2.2. Forwarding Latency. This setup can also be used to measure the forwarding latency of the switch under full bidirectional load of up to 480 Gbit/s. Load on the switch due to processing can influence the observed latency. For example, RFC 2544 requires measuring the latency of the DuT under full load [3]. The following test loads all 10 GbE ports of our switch to quantify this effect on our switch.

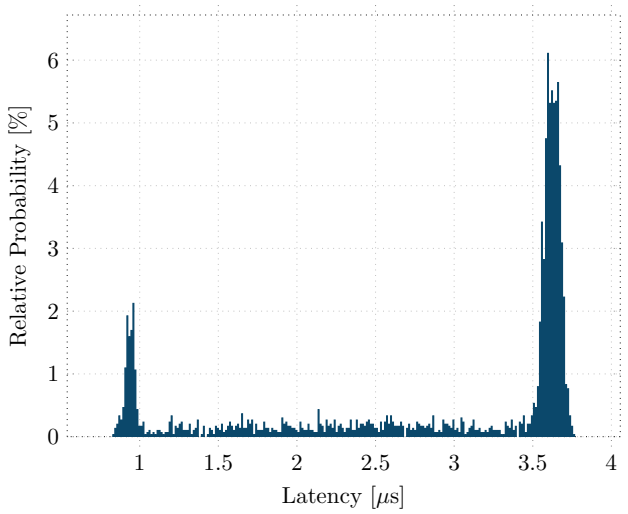


Figure 3. Latency distribution of 1 Gbit/s RT traffic with 8 Gbit/s BG traffic, QoS enabled

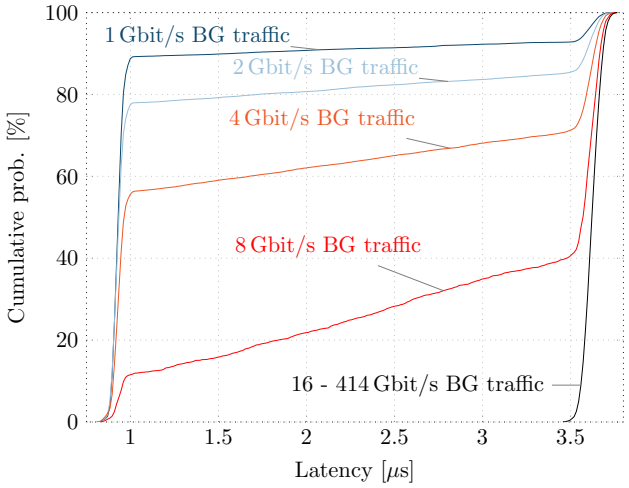


Figure 4. Latency distributions of 1 Gbit/s RT traffic with varying BG traffic, QoS enabled

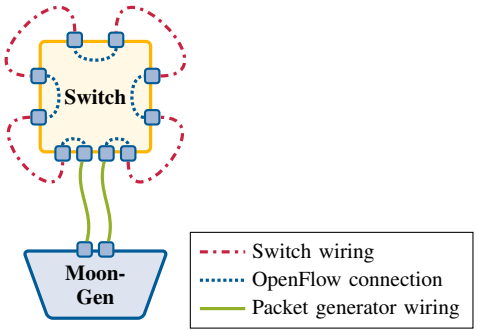


Figure 5. Loop forwarding test setup

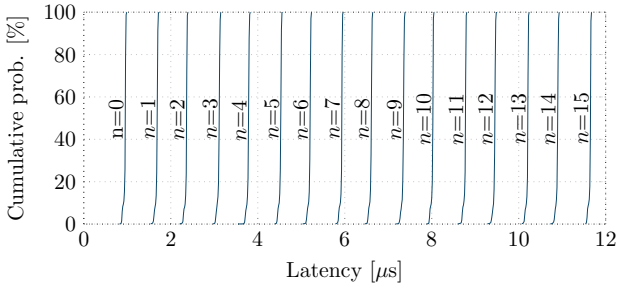


Figure 6. Latency distributions traffic forwarded through the switch n times

The total forwarding latency l consists of the delay introduced by the connection from the packet generator to the switch l_{gen}, the forwarding latency l_{switch} of the switch, and the number of hops n:

$$l = 2 \cdot l_{gen} + n \cdot l_{switch}$$

We measured the forwarding latency through the switch with various loop lengths from $n = 0$ (sending the traffic back directly) to $n = 23$. Figure 6 shows the CDFs of different loop lengths up to $n = 15$ to improve the readability of the graph as the remaining CDFs look similar. We can calculate the following median latencies from these results: $l_{gen} = 480\,ns$ and $l_{switch} = 729\,ns$. These values include propagation delay due to varying cable lengths, we used copper cables with various lengths between 0.5 and 3 meter. This introduces an additional error of 12 ns (assuming a propagation speed of $0.7c$ [7]) in addition to the granularity of 12.8 ns of the packet generator [7].

Note that these results are crucial for FLOWer: The latency of the switch is important for further tests using the switch to amplify traffic for a separate DuT. In such a setup, the switch is part of the measurement equipment, and its accuracy therefore limits the total accuracy of the experiment.

These results show that forwarding latency does not depend on the switch ports. This indicates the high accuracy of the packet generator and that latency is independent from the used switch port. We did not test all combinations of ports, one should repeat this test with the appropriate set of ports to verify this before relying on a switch to run latency-critical experiments. There may be differences in the latency between ports on a switch due to the internal architecture of the switch.

The difference between the minimum and maximum observed forwarding latency was only 217.6 ns (cf. the steep CDFs in Figure 6, each based on 48 000 timestamped packets over 48 seconds[3]). This is important when the switch is used to amplify traffic while also measuring latency, the inaccuracy of the switch affects the measurement. OpenFlow switches with a far lower jitter exist [23] and can be used if a better precision is required.

5. Amplifying Traffic

After evaluating the suitability of an OpenFlow Switch for our testing purposes in Section 4 we apply the FLOWer

3. MoonGen cannot timestamp all packets, only random samples.

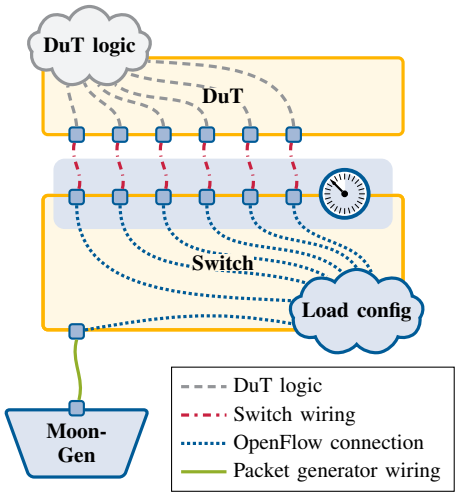


Figure 7. Testing arbitrary devices

approach to testing arbitrary networking devices. Therefore, we extend the original two-device-setup by another switch as shown in Figure 7. Both MoonGen and the DuT are connected to the switch which amplifies MoonGen's traffic and measures the throughput of the DuT. Standard tests like MoonGen's RFC 2544 [3], [24] implementation can be performed with only small modifications while increasing the bandwidth by more than an order of magnitude.

5.1. Generating Different Flows

OpenFlow flows amplify the traffic by sending incoming packets to multiple ports. Additionally, the switch can modify header fields to generate different traffic which would not be possible with traditional switches. This allows for more realistic test cases (different traffic on each port) with a multitude of network flows on different ports despite the limited output bandwidth of the packet generator. OpenFlow defines modification actions for header fields of all commonly used protocols [13] and PicOS supports most of them [18].

OpenFlow applies actions to packets, one of these actions is processing the packet via a *group* as depicted in Figure 8. Packets arrive on a port to the left, are matched via a rule and assigned to a group. Groups are meant to implement multicast and load-balancing efficiently. A group consists of multiple buckets, each defines a set of actions to be executed for packets sent to this bucket. These actions are the same that can be applied by regular rules, i.e., a group essentially clones a packet and applies different rules to each clone. Packets forwarded to the group can then be configured to be sent to either one (via hashing) or all buckets [13]. We can define a group with one bucket for each switch port and thus define modification actions that are specific to that switch port. For example, the following group table definition (in Open vSwitch syntax) shows a group that changes the IP address for each switch port, loading the DuT with completely different IP flows.

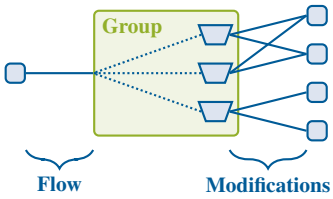


Figure 8. OpenFlow group with buckets sending to multiple output ports

```
group_id=1,type=all,
bucket=mod_nw_dst:10.0.0.1,output:1,
bucket=mod_nw_dst:10.0.0.2,output:2, ...
```

There are no restrictions for the output ports in a OpenFlow group or flow [13]. This means that a group or flow can send out a packet to a single switch port more than once by using it multiple times. This trick allows for different ratios of network flows created by the packet generator on different switch ports.

The packet generator only needs to send archetypes for various network flows which are then amplified and modified by the switch. For instance, a packet generator like MoonGen can be configured to send 5 Gbit/s TCP and 5 Gbit/s UDP traffic.

The switch can modify additional header fields to generate different traffic on different switch ports. It can even change the composition from 50:50 TCP/UDP to another ratio on some ports. For example, there can be two buckets in a group for UDP packets and none for TCP packets.

5.2. Measuring Throughput

OpenFlow counts the number of packets and bytes processed by each rule. This can be used to measure the throughput of the DuT by installing OpenFlow flows that match the traffic sent from the DuT and periodically polling their statistics. The traffic coming back from the DuT can be dropped explicitly to only count the throughput.

Statistics can be counted for multiple traffic flows by creating multiple OpenFlow flows that match on the required header fields analogous to how the switch can be used to amplify traffic flows. For example, if the switch is configured to modify the destination IP addresses for each switch port, analogous OpenFlow flows can be installed for the incoming traffic. One OpenFlow flow for each destination IP address can be installed to measure the throughput of the DuT.

Another way to measure throughput is by using OpenFlow meters. Meters are used to implement rate limiting and they also measure traffic directed to them [13]. However, a OpenFlow flow is required to send traffic to the meter and the traffic could as well be measured by the OpenFlow flow statistics. There is one scenario where this can be useful: Multiple OpenFlow flows can map to a single meter to aggregate network flows in hardware. Reading statistics can be slow, especially on switches with older CPUs [20], so this aggregation step can improve performance.

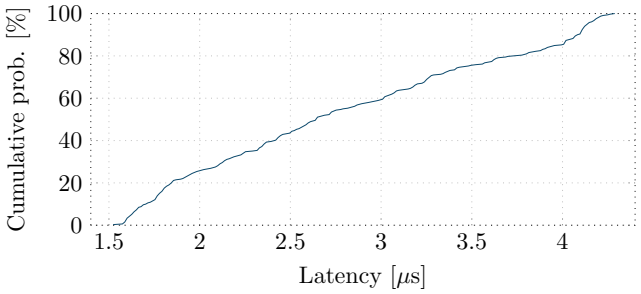


Figure 9. Latency incurred by an OpenFlow meter

Traffic can also be forwarded back to the packet generator to count it there. However, the link back is usually comparatively small here. OpenFlow meters can be used to give each network flow a fair share of the available bandwidth. This is useful to sample packets, e.g. to see if all traffic flows (e.g. wildly varying UDP ports that cannot all be installed in the hardware) get forwarded.

5.3. Measuring Latency

Latency can be measured by forwarding the packets back to the packet generator. In Section 4.2.2 we showed that the switch we used here introduces additional jitter, limiting the precision of the timestamping to ±217 ns while other switches can achieve a jitter of below 100 ns [23].

It is important to ensure no queuing happens on the amplification switch to keep the jitter at this level. The critical path is the path from the DuT back to the packet generator. Naïvely forwarding all incoming traffic to the packet generator causes queuing delay in the millisecond-range (cf. Section 4.1). The simplest solution is to filter out the timestamped packets (only a subset of the packets is timestamped) on the amplification switch and send only them back to the packet generator. A critical property of latency measurement via sampling is that the DuT cannot distinguish timestamped packets from other packets. E.g., MoonGen uses a single byte in the payload to identify timestamped packets. OpenFlow cannot match on this. However, the DuT often does not look at all header fields like the IP TTL, ECN or DSCP flags. For example, we implemented a MoonGen script[4] that marks timestamped packets by setting their TTL to 63 instead of 64. OpenFlow supports matching TTL and its exact value is usually not relevant to the DuT. Unfortunately, the switch we used does not support matching the TTL field [18], so we could not evaluate this. The actual field to hide the flag in depends on the DuT, the test case, and the capabilities of the amplification switch.

If this is not possible, e.g. due to lack of hardware support, then the only solution is using an OpenFlow meter to limit the traffic sent back to the packet generator to avoid queuing delays. Sending the traffic through a meter adds latency. Figure 9 shows the latency added by sending traffic through an OpenFlow meter which was loaded with 460 Gbit/s traffic. The difference between the minimum and maximum latency is 2.8 µs, so meters do add queuing delay, but not as much as overloading the port (cf. Section 4.1). This limits the precision of the timestamping to the µs-range.

It is of course always possible to attach the packet generator directly to some of the DuT ports, restricting the flexibility of timestamped packets. For example, MoonGen can saturate multiple 10 GbE ports, so it could load both the amplification switch and some ports of the DuT directly.

5.4. Hardware Limits

OpenFlow switches only have a limited OpenFlow flow table size. The switch we are using supports 2048 OpenFlow flows [15]. FLOWer only needs a single OpenFlow flow table entry to generate traffic and one per port to count traffic in the simplest case. More interesting scenarios need two OpenFlow flows per port and modification action: one to modify and send the packet, the other to match the modified packet and count it.

Note that the total number of different OpenFlow flows is the product of the number of modifications on the switch and the number of network flows generated by the packet generator. Modern packet generators like MoonGen allow crafting each packet in real-time through a user-controlled script, i.e., every packet can be made unique. The switch only needs one rule per output port to make every packet sent to the DuT completely unique.

The switch used here is not able to count packets on a per-flow or meter basis, only bytes. This is not a major disadvantage of this switch as the packet size is often constant or its distribution is known.

Reading statistics from the switch can also be an expensive operation on older OpenFlow switches [20]. We did not notice a significant performance degradation or CPU load when polling the statistics multiple times per second as our switch features a powerful x86 CPU.

5.5. Example: Flow Table Insertion Times

The only DuT available for this research was a second switch of the same model. A good test case for this DuT with this test setup is measuring the performance of flow table modifications under load. Using a separate DuT is a better scenario for this test than the previous self-test as the self-test requires OpenFlow flows for both the traffic generation and the tested OpenFlow flows to be on the same device. This may cause undesired interferences when attributes like insertion time or flow table size are tested. Therefore, we use the second switch as DuT here.

PicOS uses Open vSwitch to implement OpenFlow [17], so one has to understand its architecture in order to understand why this test is of particular interest.

5.5.1. Open vSwitch Architecture. Open vSwitch processes packets in two main modules: the switch daemon and

4. Available in our repository at [5]

the datapath. The former runs on the CPU of the switch and manages all OpenFlow rules. The latter performs the actual forwarding through specialized datapath rules derived on demand from the OpenFlow rules. [14]

The datapath is implemented as a kernel module in the software-version of Open vSwitch on commodity PC hardware. PicOS replaces this datapath with a specialized version that installs the rules directly in the switching ASIC [17].

This means that an installed OpenFlow flow is not automatically and instantly available as a rule in the hardware. A packet not matching any rule on the switching ASIC is forwarded to the CPU where it is processed by the switching daemon. This daemon creates a rule for the datapath (derived from an OpenFlow flow which may contact an external controller) to match future packets of the same network flow. [14]

5.5.2. Test Setup. We connected the second switch with 32 cables to the amplification switch as shown in Figure 7. The DuT was configured with OpenFlow flows matching on all combinations of the 32 switch ports and 100 different UDP ports or addresses in separate tests, i.e., a total of 3200 network flows, that forwarded the packets back to the amplification switch where they were counted via OpenFlow flow statistics (cf. Section 5.2). MoonGen was used to generate minimum-sized UDP packets alternating between 100 different UDP ports/IP addresses. This traffic was amplified 32-fold (476 Mpps at 320 Gbit/s).

We initially forwarded samples of the packets via OpenFlow meters (cf. Section 5.3) back to MoonGen to measure the point at which a rule became active with μs-level precision. However, we noted that such a high precision was not necessary for this test as the insertion times were in the order of milliseconds per OpenFlow flow. We counted the installed OpenFlow flows with two different methods: via reading the OpenFlow statistics on the amplification switch to pinpoint the time at which the first packet arrived and via reading the hardware flow table entries directly from the switching ASIC with the `ovs-appctl pica/dump-flows` command. Both methods yielded the same results in separate experiments, eliminating potential performance impacts of the `dump-flows` command and potential delays in the OpenFlow statistics.

5.5.3. Test Results. Figure 10 shows the number of OpenFlow flows installed in the switching ASIC after sending the OpenFlow commands to add the flows. Adding all of the 3200 OpenFlow flows to the flow table in the switching daemon took only 1.5 s. The number of OpenFlow rules that can actually be realized in hardware depends on the fields used by the rule. The hardware we use features 2048 TCAM entries [15]. However, each flow table entry requires two entries if all OpenFlow features are enabled. Restricting matches to layer 1 to 3 allows using 2048 OpenFlow flows [16]. No packets were forwarded for the OpenFlow flows not present in the switching ASIC as we inserted 3200 OpenFlow flows – more than the switching ASIC supports.

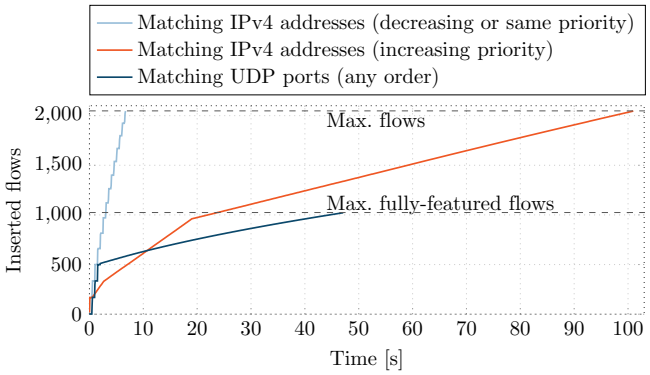


Figure 10. Flows installed in hardware over time

The insertion time may depend on the priorities of the inserted OpenFlow rules, inserting in decreasing priority is the best case and increasing the worst case [10]. We could reproduce this for rules matching on IPv4 addresses but without a significant difference between decreasing order and rules with the same priority. The insertion time was not affected by the insertion order and relatively slow for OpenFlow flows matching UDP ports. The load on the switch had no effect: We tested rates of 3.2 Gbit/s, 32 Gbit/s, and 320 Gbit/s.

The insertion time suddenly increases once the TCAM is half full at 512 or 1024 entries depending on the matched headers. Our interpretation of this result is that the hardware uses a double-buffering algorithm to speed up the required re-shuffling operations if enough space is available.

The worst-case insertion time we found was about 90 ms per OpenFlow flow for flows requiring two TCAM entries above 512 inserted flows. The best case was approximately 3 ms for IPv4 matches inserted in decreasing priority or into a half empty TCAM (6 ms for UDP port OpenFlow flows). These insertion times are significantly faster than results from older switches: Rotsos et al. found a smoothly increasing insertion time ranging from 1 s to 10 s per OpenFlow flow at 1000 flow table entries for different 1 GbE OpenFlow switches in 2012 [20]. Kuźniar et al. measured 33 to 83 ms on 1 GbE switches in 2015 [10], similar to our results.

6. Conclusions and Future Work

The results of the measurements we presented only offer incremental results over existing studies [10], [20] by using 10/40 GbE switches and vastly higher data rates. We rather introduce FLOWer as a novel approach for testing high performance network devices with minimal effort and high flexibility. This allows building high performance measurement platforms in a more cost efficient manner for either evaluating existing devices or to aid in the development of new packet processing hardware. Taking our measurement results into account, we derive the following requirements for a switch to be used with FLOWer.

- Features comparable to OpenFlow modifications including groups at line rate (some OpenFlow switches implement layer 3 or 4 matches and modifications in software, cf. [20])
- Large number of ports
- Low jitter, independent from the used port and internal packet path
- Low per-port costs, i.e., commodity hardware

Our methodology can be used with different packet generators. It is not necessarily in competition with expensive hardware packet generators. A low-end hardware packet generator can be used together with FLOWer to amplify its bandwidth while using its extensive analysis features for thousands of simultaneous network flows in hardware [21] without paying for a high-end model.

We encourage you to reproduce our measurements on your hardware. All scripts used for the experiments described in this paper and all collected data are available on GitHub [5].

In the future, we plan to use this methodology to evaluate and benchmark different devices. For this paper, only a simple SDN switch was available as DuT. More complex devices present a more interesting challenge for our methodology. In particular, we are planning to test a high-end Arbor Networks DDoS mitigation middlebox. Such DuTs are of particular interest because they are black-box software devices with high bandwidth capabilities that are beyond the reach of cheap packet generators. FLOWer puts everyone in the position to evaluate the performance of such devices without relying on expensive test equipment or vendor's promises.

Acknowledgments

This work was supported by the EUREKA-Project SASER (01BP12300A). The authors thank First Colo GmbH for providing the test hardware, Yaron Ekshtein at Pica8 for insightful discussions, and our colleagues Florian Wohlfart and Daniel Raumer for valuable contributions to this paper.

References

[1] G. Antichi, M. Shahbaz, Y. Geng, N. Zilberman, A. Covington, M. Bruyere, N. McKeown, N. Feamster, B. Felderman, M. Blott, et al. OSNT: open source network tester. *Network, IEEE*, 28(5):6–12, 2014.

[2] A. Botta, A. Dainotti, and A. Pescapé. Do you trust your software-based traffic generator? *IEEE Communications Magazine*, 48(9):158–165, 2010.

[3] S. Bradner and J. McQuaid. Benchmarking Methodology for Network Interconnect Devices. RFC 2544 (Informational), March 1999.

[4] Edge-Core Networks. AS5712-54X Datasheet. http://www.edge-core.com/temp/ec_download/1555/AS5712-54X%20ONIE%20%20DS%20add%20Cumulus%20Ready%20Logo%20R02%20.pdf. Visited: 2015-11-23.

[5] P. Emmerich. FLOWer Scripts. https://github.com/emmericp/FLOWer-scripts.

[6] P. Emmerich. MoonGen. https://github.com/emmericp/MoonGen.

[7] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle. MoonGen: A Scriptable High-Speed Packet Generator. In *Internet Measurement Conference 2015 (IMC'15)*, Tokyo, Japan, Oct. 2015.

[8] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, S. Hand, and J. Crowcroft. Queues don't matter when you can jump them! In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 1–14, Oakland, CA, May 2015. USENIX Association.

[9] IEEE. Virtual LANs. *IEEE 802.1Q-2011*, 2011.

[10] M. Kuźniar, P. Perešíni, and D. Kostić. What you need to know about sdn flow tables. In *Passive and Active Measurement*, pages 347–359. Springer, 2015.

[11] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. A Power Benchmarking Framework for Network Devices. In *IFIP Networking*, pages 795–808. Springer, 2009.

[12] Open Compute Project. Networking Specs and Design. http://www.opencompute.org/wiki/Networking/SpecsAndDesigns. Visited: 2015-11-23.

[13] Open Networking Foundation. OpenFlow Switch Specification Version 1.4.0, October 2013.

[14] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The Design and Implementation of Open vSwitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA, May 2015. USENIX Association.

[15] Pica8. Flow Scalability per Broadcom Chipset. http://www.pica8.com/document/v2.6/html/hardware-guides/#6914398. Visited: 2015-11-23.

[16] Pica8. Optimizing TCAM Usage. http://www.pica8.com/document/v2.6/html/ovs-configuration-guide/#6914964. Visited: 2015-11-23.

[17] Pica8. PicOS Overview. http://www.pica8.com/documents/pica8-whitepaper-picos-overview.pdf. Visited: 2015-11-23.

[18] Pica8. Supported OpenFlow Protocol 1.4 Features. http://www.pica8.com/document/v2.6/html/ovs-configuration-guide/. Visited: 2015-11-23.

[19] C. Rotsos, G. Antichi, M. Bruyere, P. Owezarski, and A. Moore. OFLOPS-Turbo: Testing the Next-Generation OpenFlow Switch. In *European Workshop on Software Defined Networks (EWSDN)*, 2014.

[20] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore. OFLOPS: An Open Framework for OpenFlow Switch Evaluation. In *Passive and Active Measurement*, pages 85–95. Springer, 2012.

[21] Spirent. HyperMetrics FX 2/4/8 Port 10 Gigabit Ethernet Test Module. http://www.spirent.com/~/media/Datasheets/Broadband/PAB/SpirentTestCenter/STC_HyperMetrics_fX_2-4-8-Port_10G_Ethernet_Test_Module.pdf.

[22] Spirent. HyperMetrics FX 40/100 Gigabit Test Modules. http://www.spirent.com/~/media/Datasheets/Broadband/PAB/SpirentTestCenter/STC_HyperMetrics\%20_fX_40-100G_Module_datasheet.pdf.

[23] Tolly Enterprises, LLC. Mellanox SwitchX-2 (SX1036) vs. Broadcom StrataXGS Trident II (Arista DCS-7050QX) – Performance Evaluation. http://www.mellanox.com/related-docs/products/Tolly-215111-Mellanox-SwitchX-2_Performance.pdf, 2015.

[24] P. Werneck. RFC 2544 Tests for MoonGen. https://github.com/emmericp/MoonGen/pull/98.

On Time-Stamp Accuracy of Passive Monitoring in a Container Execution Environment

Farnaz Moradi, Christofer Flinta, Andreas Johnsson, Catalin Meirosu
Cloud Technologies, Ericsson Research, Sweden
Email: {farnaz.moradi,christofer.flinta,andreas.a.johnsson,catalin.meirosu}@ericsson.com

***Abstract*—Passive monitoring of network performance parameters is experiencing a revival due to widespread adoption of virtualization and software-based implementation of network functions. Timestamping is one of the most challenging operations needed for passively monitoring network traffic performance parameters such as latency and jitter. We develop a setup whereby functions that monitor the network traffic are deployed in monitoring containers adjacently to, and interconnected through a virtual switch with the monitored Virtual Network Function instance. In this scenario, we evaluate the effects of container virtualization and virtual switch mirroring of traffic on the measurement of latency. The evaluation results indicate very low measurement errors (a few microseconds in our testbed) which are consistent over different measurement scenarios, thus validating the feasibility of this technique for passively monitoring latency.**

I. Introduction

Advances in virtualization have led to the emergence of network function virtualization (NFV) which decouples network functions from dedicated hardware to software-based applications that can run on commercial off-the-shelf hardware. Accurate, timely, and non-intrusive monitoring of network performance metrics of such virtual network functions (VNFs) is critical for identifying and avoiding violation of performance guarantees. In recent years, container-based virtualization has received considerable attention and has become a candidate for running VNFs. Container execution environments provide lightweight virtualization with much lower overhead compared to hypervisor-based virtualization with Virtual Machines (VMs), since rather than running a full operating system containers share the same kernel with the operating system of the host machine [19], [21]. Moreover, containers are created and removed much faster compared to VMs, and container-based virtualization provides better resource utilization since idle containers do not use any resources.

In Linux containers, the kernel-level namespaces are used for resource isolation and the Control groups (cgroups) [13] are used for managing and limiting resources. Cgroups also expose resource usage metrics such as memory, CPU, and block I/O which can be used for monitoring purposes. Such metrics for the containers running in a host machine can be collected by a separate container in the host, e.g., cAdvisor[1].

Existing tools for monitoring containers only provide compute resource utilization statistics as well as counters for packets and bytes on the interfaces of the containers. Network-related metrics, such as per-flow metrics can be measured by for example enabling sFlow [16] on the virtual switch infrastructure that interconnects the containers. SFlow is a general purpose monitoring tool for sampling packets and interface counters on packet forwarding devices. However, sFlow does not provide end-to-end measurements of metrics such as delay, jitter, and packet loss [2].

End-to-end measurements of network performance can be performed using active or passive measurement methods. In active measurements, test packets are injected from a probe in the network and are received in another probe. In passive measurements, in contrast, actual network traffic is being observed without injecting probe packets. In addition to network performance monitoring, passive measurement methods are also used for example to perform traffic analysis for traffic profiling, classification, and characterization, anomaly and intrusion detection, and debugging.

One option for measuring end-to-end network performance metrics in containers is to run monitoring functions inside the same container in which the VNF application is being executed. This requires the monitoring code to be added to the image of the VNF or executed inside the container after instantiation of the VNF container. Another option, which we investigate in this paper, is to execute monitoring functions in separate and adjacent monitoring containers. The monitoring container receives a copy of packets originated from or destined to the VNF instance and calculates different network performance metrics.

Running the monitoring functions in a separate container instead of running them inside the VNF container has many advantages. Some of these advantages are as follows:

1) The monitoring can be done transparently without a need to run any process inside the VNF container or instrument the VNF image with required software.
2) A single monitoring container can be used to monitor multiple VNF containers in contrast to running a monitoring process in each VNF container.
3) Using a monitoring container enables separation of the monitoring process from the VNF processes by assigning them to different CPU cores.
4) Monitoring is isolated from VNF so failure of the monitoring process will not adversely affect the VNF container.
5) A separate container for monitoring allows more flexibility in running different monitoring functions so that they can be updated, re-configured, or changed

[1] https://github.com/google/cadvisor

ISBN 978-3-901882-83-8

on-the-fly transparently to the VNFs.
6) Migration of the VNF instance can be done independently and without affecting the monitoring.
7) The monitoring container can be controlled and managed by the infrastructure provider who may be different from the VNF provider.
8) The monitoring container can create log files from the VNF traffic which can be used for debugging the application instead of the application itself generating the logs.

Despite many advantages of running monitoring functions in separate monitoring containers, the effect of such setup on the accuracy of timestamps required for passive network monitoring is not known. Previous studies have shown that hypervisor-based virtualization affects the timestamping and performance measurement accuracy [5] [14]. However, none of the previous studies have looked into the effects of container-based virtualization on the accuracy of measurements, particularly the accuracy of the timestamps required for passive network performance monitoring.

In this paper, our contributions are as follows. We develop a setup for passive monitoring of network traffic by introducing standalone monitoring containers which are interconnected through a virtual switch with the monitored VNF containers. We study the effect of container virtualization and packet copying in virtual switches on the accuracy of timestamps obtained by the monitoring functions running inside the monitoring containers. Moreover, we evaluate the accuracy of passive latency monitoring using different methods in our setup. Our measurement results indicate low and consistent timestamping errors over different measurement scenarios, therefore confirming that our monitoring setup is suitable for passive monitoring in container execution environments.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes our measurement system for passive container-based monitoring and Section IV presents our testbed and experimental settings. In Section V the evaluation method is described and the experimental results are presented. Section VI presents a discussion of our findings. Finally, Section VII concludes the paper.

II. Related Work

Existing tools for monitoring containers only gather metrics such as CPU, memory, and block I/O usage for containers running in a host machine. These metrics can be obtained from cgroups in Linux [13]. Network metrics which can be collected by existing tools are limited to the number of packets and bytes received/transmitted from an interface. Although sFlow [16] can be used for sampling packets and interface counters, it does not provide network metrics such as latency.

Active and passive measurements of network performance metrics can be performed using OpenFlow messages in OpenFlow-enabled virtual and physical switches in the network. Examples of such methods are latency measurements [17] and link utilization monitoring [23], however these methods are not in the scope of this paper since they are not designed for passive measurements for traffic analysis.

The effects of virtualization on network measurements have been previously studied. In [14] the authors studied how timestamping variability is affected by hypervisors and showed that only under certain conditions timestamping performance in virtualized environments gives good results. In [5] the effects of Kernel-based Virtual Machine (KVM) virtualization on active round trip time measurements have been evaluated. The results show that the measurements are affected by both CPU load in the host and load in the network. Although different studies have investigated the effects of virtualization on timestamp accuracy of measurements, to the best of our knowledge effects of container-based virtualization on passive measurements have not been investigated before.

A passive network monitoring function which requires accurate timestamping is latency. In the rest of this section we mention some related work for passive latency monitoring. Note that none of these methods have been designed or evaluated in virtual platforms such as VMs and containers.

A naïve way to passively measure latency is to store packet hashes together with timestamps at both the sender and receiver side and periodically exchange and compare the hash and timestamp values. However, such an approach enforces high storage and communication overhead. In order to overcome the problems of the naïve approach, a variety of efficient passive latency measurement methods have been proposed in the literature. These methods can be roughly divided into three categories: aggregate, per-flow, and per-packet methods.

In [9] an aggregate method has been proposed, where a Lossy Difference Aggregator (LDA) data structure is created at both the sender and the receiver side and at the end of each measurement interval is exchanged for calculating loss and latency. LDA requires synchronization packets to be sent over the same channel as the data traffic and makes an assumption that the packets arrive in FIFO order. FineComb [12] also calculates aggregate latency but rather than making an assumption about packet ordering proposes a stash data structure to recover from packet reordering.

Aggregate latency measurements cannot capture the latency experienced by different flows, therefore more fine-grained latency measurement methods have been proposed in the literature. Reference Latency Interpolation (RLI) [10] obtains per-flow latency measurements, however it requires injecting reference packets (probe packets). MAPLE [11] presents an architecture for latency monitoring where the granularity of the measurements can be selected. In MAPLE the focus is on delay storage and query and it is assumed that the packets can carry timestamps. In [18] COLATE, a counter-based per-flow latency estimation scheme for latency monitoring without using any probes or timestamps inside packets has been proposed. COLATE can achieve accurate results with low overhead compared to previously existing methods. More details about this method can be found in Section V-B3.

In addition to per-flow measurements more fine-grained results can be achieved using per-packet methods in exchange for extra costs. For example, in [22], an approach for per-packet delay and loss measurements has been proposed which first uses an order preserving aggregator (OPA) data structure to transmit ordering information for recovering from packet reordering and then sends the compressed packet timestamps to be used for estimating delay.

In this paper, rather than proposing a new latency monitor-

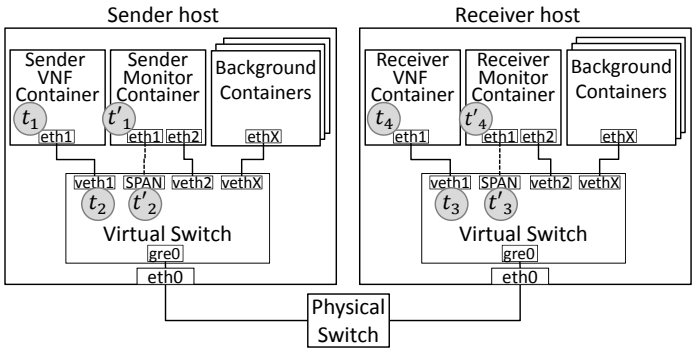


Fig. 1: Experimental setup with two host machines with Open vSwitch and Docker containers.

ing algorithm, we investigate how existing methods perform in a container execution environment. As part of the evaluations presented in our paper, we use the data structures proposed by COLATE to implement a passive latency monitoring tool running inside monitoring containers adjacently to the monitored VNF container instances.

III. Measurement System and Challenges

In this section, we present our setup for passive monitoring of VNFs and applications running in a container execution environment. In this setup, a monitoring container is instantiated and attached to the same virtual switch to which the VNF is connected. The virtual switch is then configured to duplicate the packets originated within or destined to the VNF container and send the copies to the monitoring container.

In general, duplicating and copying packets for passive network monitoring can be performed in two different ways: in-line mode and mirroring mode [7]. In the in-line mode, a network tap (a.k.a. Test Access Port (TAP)) is used to duplicate all traffic passing through it and provide a connection to the capturing device. In mirroring mode, a network switch duplicates the packets from one or more ports and sends the replicates to a single monitoring port (a.k.a. Switched Port analyzer (SPAN) port) to be captured for analysis.

The monitoring function which is executed inside the monitor container receives the copy of VNF packets on its interface. The monitoring function can then perform filtering and sampling of packets before using them for estimating different network performance metrics. In our setup, the monitoring functions in different hosts can also communicate with each other via a separate interface in order to exchange control and synchronization messages which are required for different types of measurements such as packet loss and latency.

The copying of packets can affect the accuracy of the packet timestamps. Our main focus in this paper is to investigate the accuracy of packet timestamps observed at the monitoring containers in comparison to what is observed inside the VNF containers. Therefore, we calculate timestamping error as the difference of a packet timestamp observed in the VNF and monitor containers. Figure 2 shows how the timestamping errors on the sender and the receiver hosts are calculated. Further, the effect of loading different resources on the host machines on the accuracy of the timestamps observed in the monitoring containers is investigated.

Additionally, we evaluate the effect of timestamp accuracy on latency measurements between different hosts. Latency can be passively measured by either round trip time (RTT) or one-way measurements. In passive RTT measurements, the request and reply packets of a flow are matched with each other and their timestamps are compared in order to estimate the RTT. However, passive RTT measurement is not always possible, e.g., for UDP communications. Passive one-way latency measurements can be used for any type of traffic where the timestamps of packets at the sender and the receiver sides are compared against each other.

A common assumption made by passive one-way latency measurement methods, is that the clocks at the sender and the receiver hosts are tightly synchronized, since unsynchronized clocks lead to errors in the measured values. In this paper we study the effect of running passive latency functions in separate containers and compare the results with the latency values which could be obtained from inside the VNF containers as shown in Figure 2. Errors caused by synchronization problems are not part of this study. More information about general synchronization issues and achieving high precision can be found in [3] and [20], respectively.

IV. Testbed setup

Figure 1 shows the testbed used for evaluating the container-based passive monitoring setup presented in this paper. This testbed is comprised of two physical host machines that are connected with a physical switch. Each host runs a VNF container and a monitoring container as well as a number of background containers. The virtual switches on the host machines are connected to each other with a Generic Routing Encapsulation (GRE) tunnel and each switch is responsible for tapping or mirroring the VNF packets to the SPAN port, so that a monitoring function which runs in the monitoring container can capture and analyze the packets. The background containers are used for testing purposes such as generating background CPU and network load.

In our measurements, we have used Docker containers[2] and Open vSwitch (OVS)[3] virtual switches. When a Docker container is created, a pair of peer interfaces is created where one peer becomes the interface for the container and the other one is bound to the virtual switch. OVS is an OpenFlow switch which uses flow classification and caching techniques to provide high performance forwarding [15].

In OVS, the data plane is implemented in kernel space while the control plane is in user space. OVS manages packets as flows where the first packet of a flow is sent to the controller. The controller determines how the packet should be handled and passes the packet back to the data plane. Additionally, the controller updates the cache in the kernel module so that the rest of the packets of the flow can only go through the data plane following the given instruction. This means that the first packet of a flow which has to cross the boundary of kernel and user space and be classified by the controller

[2]https://www.docker.com/
[3]http://openvswitch.org/

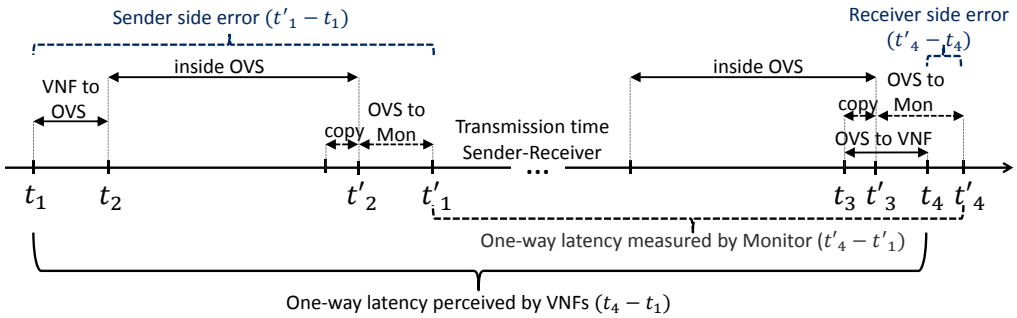


Fig. 2: Timestamp errors on sender and receiver sides.

experiences longer processing time in the switch compared to the rest of the packets. The flows in the kernel cache that have been idle for a configured amount of time are then removed by the controller [15].

Using OVS allows us to perform both tapping and mirroring of traffic. Tapping can be done by adding a flow to the OpenFlow switch to perform two actions on each packet coming from the VNF port. One action is to forward the packet normally, and the second action is to send it to the monitoring port (e.g., *ovs-ofctl in_port=1,actions=NORMAL,output:2*). Mirroring is supported by OVS which is implemented in user space through modification of the same flow table exposed through OpenFlow and can be used by adding a mirror to the virtual switch and defining the input and output ports.

In order to evaluate the effects of container-based virtualization and packet duplication on timestamp accuracies, we have used the *tcpdump* tool for obtaining packet timestamps on different capture points. Figure 1 shows the capture points, i.e., interfaces on which we used *tcpdump*. On the sender side, *tcpdump* is executed inside the VNF container to obtain timestamp t_1, on the host machine to obtain timestamps t_2 and t'_2, and inside the Monitor container to obtain t'_1. The timestamps on the receiver side are obtained similarly, i.e., t_3 and t'_3 on the host machine and t_4 and t'_4 in the VNF container and the monitor container, respectively.

In our experiments, different tools have been used to study the effects of loading resources on the measurements. The CPU, I/O, and virtual memory loads are generated using the *stress* tool [1], where we start multiple background containers running the tool. The network load on the virtual switches has been emulated by using the *tcpreplay* tool and replaying random pcap files from inside the background containers. We have also used *tcpreplay* to cause network congestion on the path between the host machines.

V. Evaluation

In this section we present our evaluation method and the experimental results. The measurements are divided into two subsections: (V-A) Timestamping errors, and (V-B) Passive latency measurements. In subsection V-A we present measurement results for three scenarios: (1) measurements with no background load, (2) measurements with loading different resources such as CPU, I/O, and virtual memory, and (3) measurements with network load on the virtual switch. In subsection V-B results for (1) passive round trip time measurements, (2) passive one-way measurements, and (3) passive one-way measurements using COLATE [18] are presented.

A. Timestamping errors

In this section we present our measurement results on comparing the timestamps of packets sent by a VNF container with the timestamps of the copy of the packets captured in the corresponding monitor container. The goal is to investigate the effects of container-based virtualization as well as switch processing and packet copying on the accuracy of the timestamps.

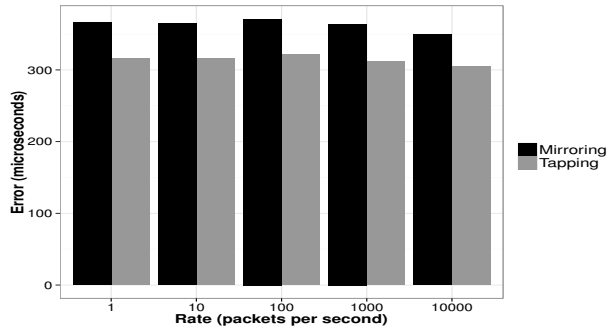
For each measurement, we emulate the VNF traffic by running the ICMP *ping* tool with different frequencies inside the sender VNF container which allows us to also measure RTT between the VNFs. The *tcpdump* tool is used both in the VNF and in the monitor container to capture the echo request messages. In our experiments, the main focus is on measuring the errors caused by container and switch virtualization, which are not dependent on the type of VNF traffic. For each packet, we calculate and report the timestamping errors. During our measurements we observed that when ping is running with higher packet send rates, it returns lower RTT values regardless of where it is running, i.e., inside the host or inside a container. For example, we observed that pinging the receiver VNF container from inside the sender VNF container with frequency of one packet per second with packet size of 64 bytes returns an average RTT of 485 μs compared to 331 μs for 1000 packet per second rate. This is in line with results observed in [5].

1) Measurements with no load:

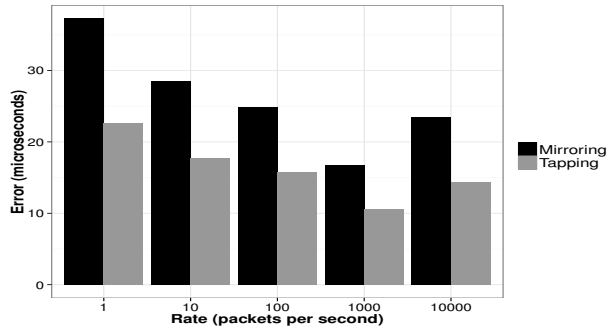
The first set of experiments is performed with no background container and no extra network load. The main goal of these measurements is to study the errors caused by the container virtualization and virtual switch processing including packet mirroring/tapping.

Figure 3 shows the timestamping errors on the sender host comparing tapping and mirroring. The y-axis shows the measurement error, where for each packet we calculated the error as the difference between the timestamps reported by *tcpdump* in the VNF and in the monitor, i.e., $t'_1 - t_1$ as shown in Figure 2. The x-axis shows the send rates given as input to the ping tool running inside the VNF container. We have observed that the ping does not send the exact number of packets with the input send rate, e.g., for input send rates 100 and 10000 packets per second, ping sent around 84 and 27000 packets per second, respectively. However, in the figures the input parameter for ping is shown, indicating the order of magnitude for the send rates.

It can be seen that the time it takes for the first packet (Figure 3a) to be received by the monitoring container is

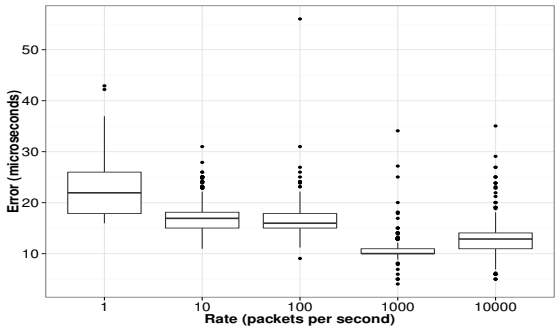


(a) First flow packet

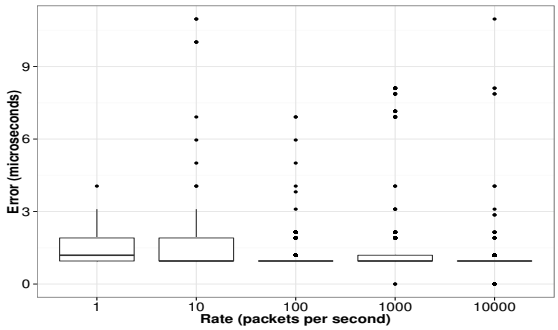


(b) Mean for the rest of the flow packets

Fig. 3: Comparison of the error caused by tapping versus mirroring packets to the monitoring container on the sender side for different send rates with no background load ($t'_1 - t_1$).



(a) Sender host ($t'_1 - t_1$)



(b) Receiver host ($t'_4 - t_4$)

Fig. 4: Comparison of the error on sender and receiver side for different send rates (excluding the first flow packet).

significantly higher than the average value for the rest of the packets (Figure 3b). On average the error was 328 μs and 375 μs for the first packet using tapping and mirroring respectively, with standard deviation of 28 μs and 25 μs. For the rest of the packets in each flow the error was on average 16 μs and 26 μs with standard deviation of 2 μs and 4 μs using tapping and mirroring respectively. As mentioned in Section IV, for each new flow OVS sends the first packet to the controller in the user space and the rest of the packets are forwarded in the data plane in the kernel space. Moreover, it can be seen that the timestamps obtained from tapping of packets are slightly closer to the timestamps perceived by the VNF container compared to mirroring, i.e., smaller error values. In the rest of our measurements we only use tapping because of the lower error rate. The figure also shows that the error for lower send rates, particularly one packet per second, is higher than the measurements with higher send rates. As mentioned above, ping returns higher RTT values for low send rates which also affects the measurement results and is the reason for higher errors shown in the figure.

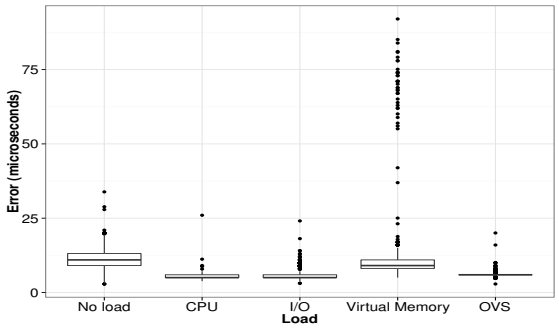
Figure 4 shows the timestamping error on the sender and the receiver hosts for different send rates. It can be seen that the errors on the sender side are much higher than the errors on the receiver side. The main reason for the differences, as shown in Figure 2, is that on the sender side each packet is first observed in the VNF (t_1) and then enters the switch (t_2) where it is processed and then is duplicated and its copy is sent to the monitor interface (t'_2) and is observed by the monitoring function (t'_1), while on the receiver side the switch processing time affects each packet and its copy similarly. Additionally, in the receiver the time it takes for the packets to be observed inside the VNF (from t_3 to t_4) and the receiver monitor container (from t'_3 to t'_4) are equal, and the main factor that causes errors is the packet duplication time in the switch which is required for both tapping and mirroring.

In our measurements, we observed that running *tcpdump* in all the four capture points inside the sender host increases the errors compared to the experiments where *tcpdump* was only executed inside the containers. For example, we observed that when *tcpdump* was capturing traffic on all the measurement points on the sender side, the total timestamping error was around 14 μs higher than the measurements where *tcpdump* was only running inside the containers. This observation shows that *tcpdump* also affects the accuracy of our measurements.

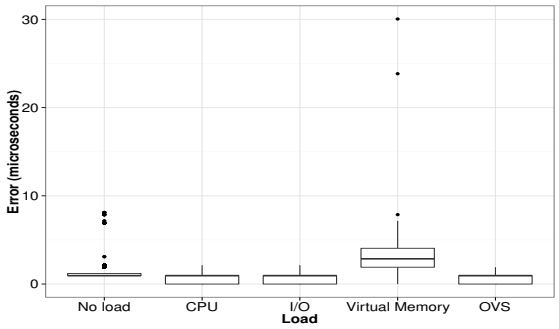
In order to estimate the time it takes for packets to reach the OVS interface from the interface inside the container and vice versa without running *tcpdump* in all four capture points, we performed the following measurements. We calculated the values for $t'_2 - t_2$ when *tcpdump* was only running on the host as well as the values for $t'_1 - t_1$ when *tcpdump* was only running inside the containers. The values obtained from the capture points in the host were subtracted from the values obtained from the containers to estimate the time $(t_2 - t_1) + (t'_1 - t'_2)$. The results indicate that the time it takes for packets to arrive from the VNF container to the OVS and from the OVS to the monitor container is less than 0.8 μs in average with standard deviation 0.5.

2) Effect of Resource load:

In this section we study the effect of loading physical



(a) Sender host ($t'_1 - t_1$)



(b) Receiver host ($t'_4 - t_4$)

Fig. 5: The effect of high loads on host resources on per-packet timestamping errors excluding the first packet.

resources on the host machines on the accuracy of the packet timestamps obtained by the monitoring containers. In order to load the resources on the host machine, we started eight background containers running the *stress* tool to load different resources. We used the *stress* tool to spawn a given number of workers spinning on *sqrt()* to load the CPU, *sync()* to load I/O, and *malloc()*/*free()* to load the virtual memory.

Figure 5 shows the results of loading CPU, I/O, and virtual memory. It can be seen that loading memory increases the error caused by copying packets on both the sender and the receiver sides, while loading CPU or I/O in average reduces the time compared to measurements with no background load.

The main reason for reduced error when physical resources including CPU are loaded is due to CPU auto scaling which happens when the system is under load. By default the CPU governor is set to "ondemand" which allows CPU to achieve maximum clock frequency when the load is high and achieve minimum frequency when the system is idle which allows the system to adjust power consumption according to system load.

3) Effect of network load in virtual switch:

In this section we study the effect of network load in the virtual switch on the accuracy of the packet timestamps. We used the *tcpreplay* tool to load the virtual switch with network traffic. In the measurements, eight background containers were attached to the OVS and replayed traffic from a given pcap file with maximum possible speed. The traffic was forwarded to a 9th background container by the OVS in the host where the packets were received and dropped. During these experiments, the CPU on the host machine also reached maximum load but we did not observe any ICMP packet loss.

Figure 5 shows the results for this scenario with network load in the virtual switch (OVS). It can be seen that the error caused by copying packets is very similar to the experiments with CPU loading.

B. Passive latency measurements

Latency monitoring requires accurate timestamping. In this section we evaluate the effect of our passive container-based monitoring setup on the accuracy of round trip time (RTT) measurements and one-way latency measurements.

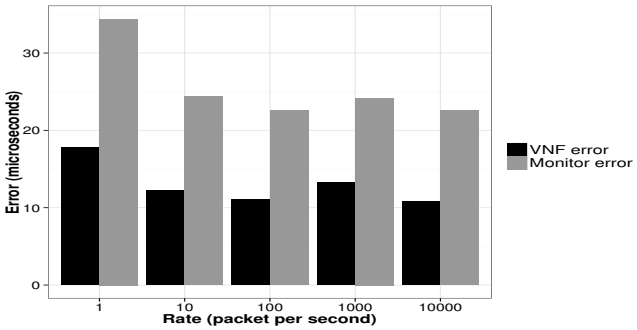
For RTT measurements, we use the values obtained from ICMP pings, which were sent from inside the sender VNF container to the receiver VNF container, as the ground truth since it is the actual RTT experienced by the VNFs. Then we evaluate the measurement results from both inside the VNF container and the monitor container on the sender side. For passive latency measurements, we use the latency values perceived by VNF containers as the ground truth and evaluate the accuracy of the latency values reported by the monitoring containers. Even though the ground truth one-way latency values are affected by the precision of time synchronization, the synchronization affects the latency calculated by the monitor containers similarly, and therefore it does not affect the reported error values. Moreover, we compare the values obtained from the above naïve approach with the latency values measured by a state-of-the-art passive monitoring algorithm which is implemented by using the data structures and the average latency estimation method presented in [18].

The measurements are done for two scenarios: with no background traffic and with network load which causes congestion on the path between the host machines. In these measurements a background container in the sender host sends packets with a high rate and the traffic is received and dropped by the receiver host. The high send rate causes network congestion and leads to VNF packet loss. However, the loss rates did not change with different VNF send rates.

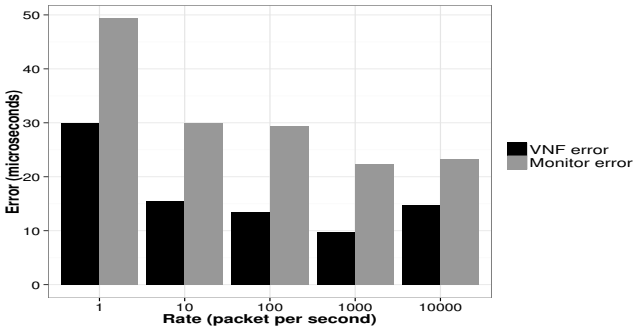
1) Passive Round Trip Time:

The RTT for each packet as perceived by the VNF containers is calculated by comparing the timestamps of each *echo request* and *echo reply* packets with the same sequence number, (i.e., $RTT_{VNF} = t_1(reply) - t_1(request)$) where $t_1(request)$ is the time when an echo request packet was sent from the sender VNF container and $t_1(reply)$ is the time that the echo reply packet corresponding to that request is received by the sender VNF container. Similarly the RTT values as perceived by the sender monitor container are calculated as $RTT_{Mon} = t'_1(reply) - t'_1(request)$. In our measurements, the average RTT reported by ping for different send rates was 359.8 μs compared to 346.8 μs and 334.2 μs as reported by the VNF and the monitor containers, respectively.

Figure 6 shows the measurement errors of the RTT measurements for VNF and monitor container versus the ground truth, i.e., values returned by ping. It can be seen that even the RTT values reported by the VNF container using *tcpdump* are underestimated compared to the ground truth. The reason for underestimation is the delay for the request packet to be sent from the ping application to the interface where *tcpdump* records a timestamp, as well as the delay for the arriving reply

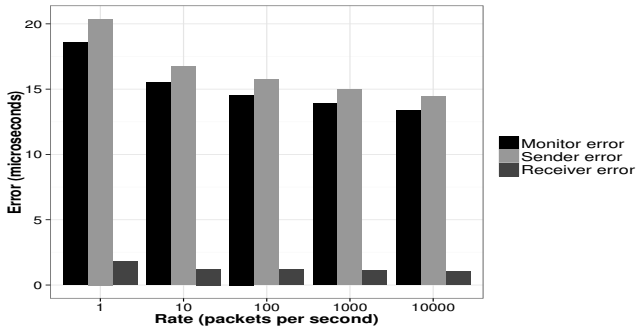


(a) No load

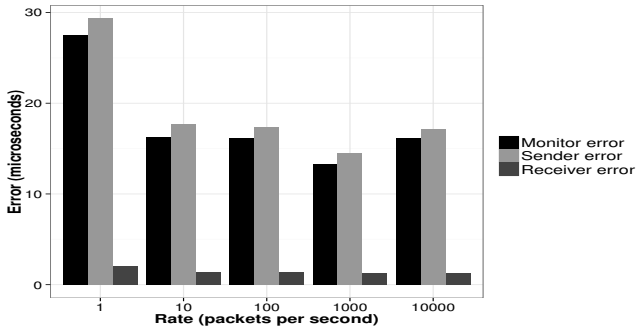


(b) Network Congestion

Fig. 6: Error of RTT measurements calculated from inside the VNF and monitor containers compared to the ground truth values reported by ping, i.e., VNF error ($RTT_{ping} - RTT_{VNF}$) and monitor error ($RTT_{ping} - RTT_{Mon}$).



(a) No load



(b) Network Congestion

Fig. 7: Error of latency measurements calculated from inside the VNF containers and from inside the monitor containers ($L_{VNF} - L_{Mon}$) in comparison with the timestamping errors on the sender side ($t'_1 - t_1$) and the receiver side ($t'_4 - t_4$).

that is first timestamped by *tcpdump* and then by the ping tool. It should be noted that the errors of the values reported by the monitoring container compared to the ground truth also include the VNF errors. Overall, it can be seen that the errors of the measurements in the monitoring container are consistent for different measurements. Moreover, even though in experiments with network congestion 63% of packets were lost and the RTT reported by ping was increased to 28.5 ms, the error values were only slightly increased.

2) Passive latency:

The one-way latency for each packet as perceived by the VNF containers is calculated as $L_{VNF} = t_4 - t_1$ and is used as the ground truth. The latency values as perceived by the monitor containers are calculated as $L_{Mon} = t'_4 - t'_1$. The actual values reported by one-way latency measurements are not accurate due to drifting clocks (we have used NTP for time synchronization which does not provide the required microsecond synchronization precision for our measurements). Nevertheless, the obtained passive one-way measurement values can be used for identifying latency changes over time and detecting congestion in the network.

Figure 7 shows the error values for one-way latency measurements. Our measurements show that the latency values calculated from inside the VNF containers and from inside the monitor containers are very close to each other and the measured latency is only slightly underestimated, in average around 15 μs. These results are in line with what was observed for RTT measurements, where the absolute average error for values reported from the monitor and the VNF was around 13 μs. The figure also shows the sender and the receiver side timestamping errors. It can be seen that the errors on the sender side are much higher than the errors on the receiver side. As depicted in Figure 2, the total error of monitored one-way latency values are caused by the error which happens due to packets arriving in the sender monitor after the copying and processing times on the sender side and the error which happens due to packets arriving in the receiver monitor after being copied in the switch.

In the measurements with network overload, the heavy load on the link caused in average 62% packet loss for the VNF traffic in our measurements. The loss rates were caused by the background traffic and did not change for different VNF send rates. Figure 7 shows that the increase in network load causes the average per packet latency error to only slightly increase compared to the tests with no background network load.

3) Passive latency measurements with COLATE:

In the measurements presented in the previous sections, every single packet has been compared on the sender and receiver side. However, such naïve approach is not suitable for automatic and continuous passive latency measurements due to high storage and communication cost.

COLATE [18] is a counter-based per-flow latency estimation scheme which "allows noise in recording times for minimizing storage space, and then statically de-noises the recorded information for obtaining accurate latency estimates". The COLATE scheme consists of two phases: recording and

querying. In the recording phase, for each packet at the measurement point, a timestamp is recorded using a vector called counter vector. Moreover, each unique flow is mapped to a unique subset of these counters called counter subvectors. In other words, each packet is first mapped randomly to a counter in the counter subvector of its corresponding flow, and then the current time is added to that counter.

The average latency for flow f in COLATE is estimated as $\tilde{\mu}_f = 1/p_f(\tilde{t}_{f,R} - \tilde{t}_{f,S})$, where $\tilde{t}_{f,X} = 1/(n-m)\{n\Sigma_{j=1}^{m} S_X^f[j] - mt_X\}$ is the estimated sum of all timestamps of the packets in flow f which is calculated both at the sender ($\tilde{t}_{f,S}$) and the receiver ($\tilde{t}_{f,R}$). In these equations, S_X^f denotes the counter subvector of flow f, n denotes the number of counters in the counter vector C_X, m denotes the number of counters in S_X^f, t_X denotes the sum of all counters in C_X, and p_f denotes the number of packets in flow f.

In [18], it is shown that COLATE achieves high accuracy compared to other existing methods using simulations and real network traffic. In this study we use COLATE as an example of an accurate and low-overhead passive latency measurement method and investigate the effect of our monitoring setup on its accuracy.

We have implemented the COLATE method to be executed in measurement sessions instead of the original design with two separate recording and querying phases. In the start of each session, the sender monitor function sends a start message to the receiver and starts collecting packets using *libpcap* library. The receiver side similarly starts passive data collection after receiving the start message. At the end of the interval, the sender sends a stop message, together with the estimated sum of all timestamps $\tilde{t}_{f,S}$. Upon arrival of the stop message, the receiver compares its estimated sum of all timestamps $\tilde{t}_{f,R}$ with the value received in the message. The difference gives us the estimated average latency over the measurement session. In our experiments we only run tests for a single flow measurement in the container-based setup. In the original design of COLATE, the number of packets per flow p_f is obtained from a separate tool. In our implementation this value is also calculated by the monitor function itself. Moreover, in our implementation the stop message also carries an incremental stream digest value, i.e., a hash value created from all the packets in the measurement session, which allows us to make sure that the same set of packets are being compared in each session.

Figure 8 shows the one-way latency errors for different measurements with different session lengths. The x-axis shows the number of packets that were collected in each session. The bars in the figure show the errors between the average latency value calculated by COLATE and the average latency from per-packet measurements from inside the monitor container ($L_{COL} - L_{Mon}$). It can be seen that the error of measurements performed by COLATE depends on the number of packets in each measurement session. Our measurements show that COLATE slightly overestimates the latency values compared to per-packet values obtained from the monitoring container. Overall, one should take into consideration that the total error of measurements using this algorithm is the sum of the overestimation of COLATE and underestimation caused by using our monitoring setup with adjacent monitoring containers.

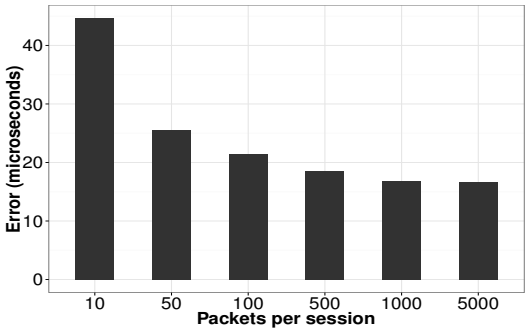


Fig. 8: Error values for average latency by using COLATE compared to a naïve method with different session lengths.

VI. Discussion

The monitoring setup presented in this paper can be used for different types of measurements. Some measurements such as packet loss detection do not require accurate timestamping. Other types of measurements such as available capacity measurements [4] require calculating the time difference between packets. Further, passive latency calculation is a challenging problem which also requires accurate timestamping. In this study we have evaluated a container-based measurement setup and how it affects the accuracy of passive latency estimation.

In this paper, we have performed different measurement experiments to evaluate the accuracy of measurements in a setup corresponding to real-world deployment of VNFs. In the measurements, ICMP ping was used to emulate VNF traffic. We also observed similar measurement results when the VNF containers where sending UDP traffic. This observation confirms that the type of traffic being passively monitored does not affect the accuracy of the timestamps in our monitoring setup. A future study could be performed with different packet sizes and random send intervals to get a more detailed view on the impact from packet size and send patterns on the timestamping accuracy. We have also shown that the main cause of error in timestamping information is due to switch packet processing on the sender side. It is expected that by continuous enhancement and optimization of virtual switch implementations, these errors caused by switch processing will be reduced. Our measurements have also shown that under heavy CPU loads, the timestamping errors caused by both container and switch virtualization are reduced. The load on the CPU has also caused the ping tool to report lower RTT values compared to experiments with no background load. These observations are due to CPU auto scaling which increases the clock frequency of CPU on the host machine. This means that by increasing the clock frequency of CPU, it is possible to reduce the timestamping errors and achieve more accurate results in expense of losing power saving benefits. Additionally, we observed that pinning the monitoring container to a CPU core can reduce the measurement errors, particularly in measurement scenarios where virtual memory is loaded.

In previous studies it has been shown that port mirroring has a number of drawbacks such as adding additional burden on the CPU of the switch [8]. In our experiments the extra CPU burden added by packet duplication was negligible. Moreover, it was shown in [24] that mirroring may introduce delay, loss,

and reordering of packets due to buffering of packets before forwarding them. In our measurements we did not observe any packet loss due to mirroring or tapping. However, we observed re-ordering of the packets received by the monitoring container compared to the packets sent from a VNF container both for the tapping and mirroring experiments. The re-ordering happens among the packets that belong to different flows and not the packets that belong to the same flow. Therefore, if the monitoring function or the traffic analysis tool requires accurate ordering of the packets for different flows, using mirroring/tapping in OVS can lead to inaccurate results.

An interesting future direction would be to evaluate the performance of different types of virtual switches. Initial experiments comparing OVS with original Docker bridge has shown that although the time it takes for the first packet to be processed is higher than the rest of the packets in both of the switches, the processing time for OVS is much higher. Overall, the average processing time it takes for all packets of a flow to be processed by the Docker bridge is lower that the OVS. Additionally, Docker allows a container to share the network stack of another container instead of directly connecting to the bridge. This means that a monitoring container can be connected to the network stack of a VNF container and perform passive measurements. The advantage of such implementation is that the errors caused by processing and copying packets by virtual switch are eliminated. However, this approach to implementation has many disadvantages including violating isolation and separation of containers.

Overall, the timestamping error caused by our setup is very low compared to the latency values reported in real datacenters. The latency values reported in [6] show that in a normal working day without any network incidents, the 50th percentile intra-pod and inter-pod round-trip latencies (from physical servers and not through virtualization layers) are around 216 μs and 268 μs, respectively. These observations also confirm the feasibility of using our setup for passive latency monitoring.

VII. Conclusions

In this paper we presented and evaluated a container-based monitoring setup for passive monitoring of VNF traffic. Our main focus has been on the timestamping accuracy which is required for passive monitoring of different network performance metrics such as latency. The key finding of our study is that the main source of error for passive container-based monitoring is caused by the switch processing time on the sender side. Moreover, we observed that the errors are quite stable and are not affected much by the load on the host machines and by congestion on the network. The results indicate that by doing initial tests and obtaining error estimates, one can calibrate a monitoring system.

References

[1] Stress tool, [online]http://people.seas.harvard.edu/~apw/stress/.

[2] Internet protocol data communication service-ip packet transfer and availability performance parameters, itu-t recommendation Y.1540, 2011.

[3] T. Broomhead, L. Cremean, J. Ridoux, and D. Veitch. Virtualize everything but time. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 1–6, 2010.

[4] S. S. Chaudhari and R. C. Biradar. Survey of bandwidth estimation techniques in communication networks. *Wirel. Pers. Commun.*, 83(2):1425–1476, July 2015.

[5] R. Dantas, D. Sadok, C. Flinta, and A. Johnsson. Kvm virtualization impact on active round-trip time measurements. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 810–813, May 2015.

[6] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *ACM SIGCOMM Conference*, pages 139–152. ACM, 2015.

[7] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *Communications Surveys Tutorials, IEEE*, 16(4):2037–2064, Fourthquarter 2014.

[8] W. John, S. Tafvelin, and T. Olovsson. Review: Passive internet measurement: Overview and guidelines based on experiences. *Comput. Commun.*, 33(5):533–550, Mar. 2010.

[9] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese. Every microsecond counts: Tracking fine-grain latencies with a lossy difference aggregator. In *Proceedings of the ACM SIGCOMM Conference*, pages 255–266. ACM, 2009.

[10] M. Lee, N. Duffield, and R. R. Kompella. Not all microseconds are equal: Fine-grained per-flow measurements with reference latency interpolation. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 27–38. ACM, 2010.

[11] M. Lee, N. Duffield, and R. R. Kompella. Maple: A scalable architecture for maintaining packet latency measurements. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, pages 101–114. ACM, 2012.

[12] M. Lee, S. Goldberg, R. R. Kompella, and G. Varghese. Fine-grained latency and loss measurements in the presence of reordering. *SIGMETRICS Perform. Eval. Rev.*, 39(1):289–300, June 2011.

[13] P. Menage. Cgroups, [online]https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt.

[14] R. Mutia, N. Sadeque, J. Andersson, and A. Johnsson. Time-stamping accuracy in virtualized environments. In *13th International Conference on Advanced Communication Technology*, pages 475–480, Feb 2011.

[15] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In *Proceedings of the 12th USENIX NSDI Conference*, pages 117–130, 2015.

[16] P. Phaal. sflow version 5, july 2004, [online]http://sflow.org/sflow_version_5.txt.

[17] K. Phemius and M. Bouet. Monitoring latency with openflow. In *Network and Service Management (CNSM), 2013 9th International Conference on*, pages 122–125, Oct 2013.

[18] M. Shahzad and A. X. Liu. Noise can help: Accurate and efficient per-flow latency measurement without packet probing and time stamping. In *the ACM SIGMETRICS Conference*, pages 207–219. ACM, 2014.

[19] Soltesz, Stephen and Pötzl, Herbert and Fiuczynski, Marc E. and Bavier, Andy and Peterson, Larry. Container-based operating system virtualization. *ACM SIGOPS Operating Systems Review*, 41:275, 2007.

[20] I. Song. IEEE 1588 and PTP, [online]http://events.linuxfoundation.org/sites/events/files/slides/elc_insop_2015.pdf.

[21] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. An updated performance comparison of virtual machines and linux containers. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015.

[22] J. Wang, S. Lian, W. Dong, Y. Liu, and X.-Y. Li. Every packet counts: Fine-grained delay and loss measurement with reordering. In *Proceedings of the IEEE 22nd International Conference on Network Protocols (ICNP)*, pages 95–106. IEEE, 2014.

[23] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha. Flowsense: Monitoring network utilization with zero measurement cost. In *Proceedings of the 14th International Conference on Passive and Active Measurement*, pages 31–41, 2013.

[24] J. Zhang and A. Moore. Traffic trace artifacts due to monitoring via port mirroring. In *Workshop on End-to-End Monitoring Techniques and Services*, pages 1–8, Yearly 2007.

Congestion Aware Priority Flow Control in Data Center Networks

Serhat Nazim Avci, Zhenjiang Li, Fangping Liu
Futurewei Technologies
{serhat.avci,andy.z.li,julius.f.liu}@huawei.com

***Abstract*—The data center bridging (DCB) protocols enable Ethernet to become the leading unified fabric for a converged data center. As a part of DCB, priority flow control (PFC) is a control mechanism that provides the losslessness feature required by certain applications such as Fibre Channel over Ethernet. Quantized congestion notification (QCN) is also a DCB protocol and provides congestion control functionality to complement PFC. However, in today's practical data center networks, QCN is not feasible and PFC is not efficient against congestion. We propose a new link layer mechanism called congestion aware priority flow control (CaPFC) that equips PFC with better congestion control capability in the absence of QCN. Changes brought by CaPFC on top of PFC are lightweight. By conducting simulations in NS-3, we demonstrate that CaPFC handles persistent and flash congestion much better than PFC. For the short messaging traffic and the query traffic, which are sensitive to tail latencies, CaPFC consistently outperforms PFC in terms of tail flow completion time.**

I. Introduction

One of the major trends in data center networking is the fabric convergence, which helps to eliminate separate networks for servers, storage and computing, thereby reducing the costs related to maintenance, management, training, equipment, cabling, power and space [1]. Data center bridging (DCB) is an effort by the IEEE 802.1 working group to promote Ethernet as the unified fabric of data center networks by expanding its networking and management capabilities, which is also called as Converged Enhanced Ethernet. DCB requires per-priority link level flow control, traffic differentiation, congestion management and transmission scheduling features to make Ethernet a candidate for unified data center networking fabric [2]. For that purpose, DCB includes four different protocols, namely priority-based flow control (PFC), enhanced transmission selection (ETS), congestion notification (QCN) and data center bridging exchange protocol.

Although the other two protocols are also important to DCB, only PFC and QCN play a role in the congestion control of the flows, which is the focus of this paper. The goal of PFC is to push the congestion from inside the network to the edges by recursively propagating this congestion to upstream nodes. In theory, PFC is envisioned to keep flow control for different priorities independent and hence create virtual lanes. Congestion created by one priority pauses only the traffic of that specific priority and does not affect the traffic of other priorities. QCN is developed to carry out large scale persistent congestion management in Ethernet-based data center networks.

PFC and QCN have proved to be successful to a certain extent. However, they have practical issues that limit their deployment. In one hand, even though PFC manages to prevent packet losses under certain assumptions, it has certain drawbacks such as congestion spread, head of the line (HOL) blocking,and potential deadlocks.

In another hand, despite the necessity of a congestion control mechanism that will complement PFC in datacenter networks and the proven performance of QCN in testbeds, QCN has serious issues that prevent its wide deployment. First of all, the scope of QCN is restricted to a single VLAN. It cannot pass through network borders [3] in data centers or Fibre Channel Forwarders (FCF) in Fibre Channel over Ethernet networks [4]. Second, QCN increases the switch and adapter complexity [2]. In a DCB switch, hundreds of congestion points are needed in addition to the tens to thousands rate limiter units in the adapters. They put a burden on the network devices in terms of complexity, power and cost. Third, QCN lacks application control, which is also pointed by [2]. As increasingly more applications prefer to control the rate at end-hosts, the QCN reaction point is disabled by default in order to prevent conflict with the rate control of applications.

This paper has three main contributions.

- We show how some overlooked assumptions about PFC fail in most merchant silicon data center switches. We show how PFC fails to prevent packet drops between ingress and egress queues in a typical merchant silicon switch. When trying to fix this issue with the current architecture of PFC, we also demonstrate that backpressure caused by PFC results in congestion spread not only among different ports and switches but also within ideally independent priorities.
- We propose congestion aware priority flow control (CaPFC) as a novel link layer mechanism that incorporates congestion control functionality in PFC. CaPFC is designed to address the shortcomings of PFC in the absence QCN. It employs a joint ingress and egress queue monitoring mechanism to handle congestion proactively.
- We demonstrate that two versions of CaPFC consistently outperforms two versions of PFC in terms of tail flow completion time. For that purpose, we build a simulation setup in NS-3 to evaluate the performance of the proposed CaPFC compared to PFC.

In the next section, we provide the technical details on the research problem and describe the proposed CaPFC mechanism. Next, we discuss how to tune the parameters in CaPFC

ISBN 978-3-901882-83-8

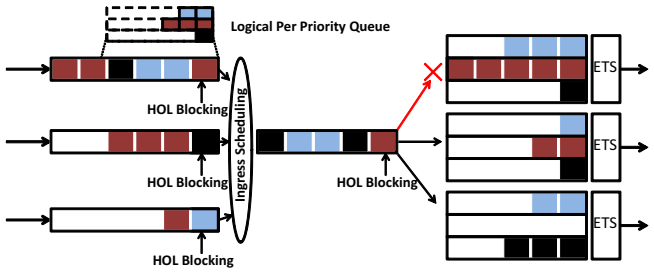


Fig. 1. The switch architecture and the HOL blocking

in Section III. The simulation setup and the simulation results are presented in Section IV. In Section V, a brief summary of the related work is presented before concluding the paper.

II. Congestion Aware Priority Flow Control

In this section, we present the proposed CaPFC mechanism and how it is integrated into the legacy PFC architecture. Before that, a typical merchant silicon architecture is presented to define the research problem and the motivation.

A. Problem Definition

The architecture of the switches that are commonly deployed in today's data center networks has a critical impact on the design and performance of the flow and congestion control mechanisms. One of the key insights behind this paper is the inefficient propagation of congestion from the egress ports to the ingress ports of a network device. The way PFC messages propagated from inside the network to the edge hosts is not very clearly analyzed. The usual assumption is that when an egress buffer of a paused port gets filled up, the packets starts to fill ingress buffers, hence PFC messages are created to the upstream device. However, there is no clear mechanism in PFC standards on how to stop switching packets from ingress to egress if there is no space left at the egress.

The reference merchant silicon switch architecture has a pipelined packet processor and a traffic manager. Whenever a packet is admitted from an ingress port, first, its header is extracted. The header is sent to a pipeline to decide what to do with the packet while the payload is stored in the ingress buffers or in shared memory buffers depending on the type of the switch. At the ingress ports, even though packets are processed on a FIFO order, we logically implement a separate queue per priority by keeping per-priority packet counters. PFC generates pause (XOFF) and unpause (XON) messages per-priority with respect to counter of each priority. When admitting packets to the pipeline, round robin scheduling is implemented between different input ports.

Unlike crossbar type of architectures, the packets are put into the egress queue whenever a forwarding decision is made, which is only after the packet processing phase. Therefore, once a forwarding decision is made for a packet, it will either will be placed at the corresponding egress queue if there is any empty space or it will be dropped. This has a sinister effect on the data transmission between ports when one of the egress queues for a single priority is completely full. The only way to prevent packet drop is to stop packet processing which will halt the data transmission towards any egress port for any priority. In this case, the packet belonging to other priorities or designated to different ports will suffer from the HOL blocking as shown in Fig. 1. In most of the merchant silicon switches, it is not feasible to stop packet processing [5] which means PFC cannot completely prevent packet drops. Dropping packets results in lengthy timeouts and retransmission and aggravates the latency and the congestion in the network. In Section IV, congestion spread, packet drops, and HOL among different priorities are empirically observed to cause high tail latencies for short flows.

This issue exists also in shared memory merchant silicon switches. In these switches, the payload is buffered only once, therefore moving a packet from ingress to egress does not occupy extra buffer space. However, the header of the packet is queued at the corresponding egress port when a decision is made. In most merchant silicon switches, the egress queues have queuing and scheduling policies to satisfy the quality of service which limit the capacity of the queues [5]. When an egress queue reaches that limit, a packet is dropped from ingress to egress even in shared memory switches.

Even though PFC is envisioned to keep flow control for different priorities independent and hence create virtual lanes, in typical switches, egress buffer overflow in one priority causes either HOL at all ingress ports for every priority or results in packet drops at the end of packet processing. Eventually, it causes a saturation tree inside the network in less than a few round-trip times, [6], which makes software solutions too slow to succeed forcing the adoption of hardware solutions. Both of these scenarios dramatically increase the tail completion time of short and time-sensitive flows. To prevent these scenarios, the key idea behind CaPFC is to develop a proactive congestion aware flow control mechanism which will keep the buffers on the egress ports from being completely occupied. In CaPFC, the virtual lanes for each priority are preserved by preventing congestion in one priority spreading to other priorities.

B. CaPFC Introduction

In typical PFC, ingress queue states are taken into account when deciding the state of flow control. CaPFC also monitors egress queues in order to add congestion control awareness on top of PFC. CaPFC employs per-input-port counters at each egress buffer to measure the congestion contribution of each input port to that specific egress buffer. It is important that these measurements are fresh and do not increase complexity as we describe in Section II-D.

In Fig. 2, it is shown that different ingress queues may send packets to a single egress queue of the same priority. If the buffer occupancy of an egress queue for a specific priority passes a certain threshold, the queue starts to count the number of packets newly arriving from each input interface. If the buffer occupancy passes the congestion threshold (explained in Section II-D), CaPFC finds the highest contributor or highest contributors of that congestion among ingress interfaces. After identifying those culprit ingress interfaces, the egress queue ask those ingress interfaces to issue PFC-XOFF messages to their corresponding upstream devices. It should be noted that CaPFC never pauses ingress scheduling for any ingress queue instead it asks them to pause the traffic from their upstream devices. The details of the CaPFC operation is explained in the following subsections.

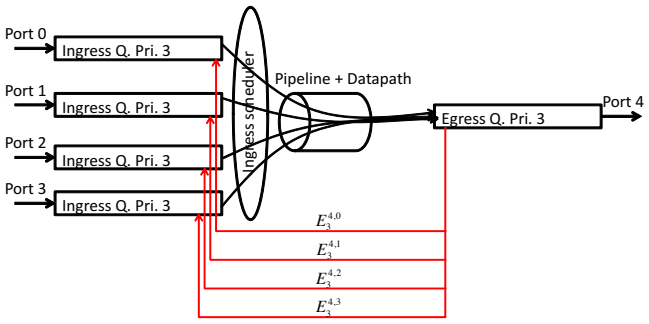


Fig. 2. An egress queue of a priority receives packets from ingress queues of other ports with that priority. In return, it sends congestion state signals per-ingress queue per-priority.

C. Ingress Congestion Management

CaPFC ingress queue management is the same as it is in PFC. CaPFC monitors the buffer occupancy of ingress queues which are PFC-enabled. Per-priority ingress buffer state keeps track of the ingress congestion behaviour. Buffer occupancy over time of an ingress queue of port i for priority p is B_p^i and it is shown in Fig. 3. MAX threshold is the maximum capacity of the queue. The variable ${I_p}^i$ keeps ingress queue congestion state of port i for priority p. It is equal to 1 in congestion (XOFF) state and 0 in non-congestion (XON) state. In the beginning, XON is the default state. In Fig. 3, when the buffer occupancy passes the XOFF threshold, this queue switches to XOFF state which is shown by dashed red line. The queue turns back to XON state once the buffer occupancy drops below the XON threshold.

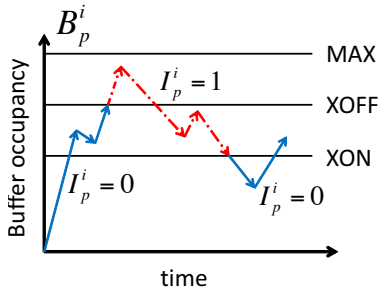


Fig. 3. Buffer occupancy over time for ingress queue buffers.

D. Egress Congestion Management

Buffer occupancy of the egress queues are monitored as in QCN. The goal is to detect congestion in the egress queues and identify the heavy flows that contribute most to the congestion. In Fig. 4, the buffer occupancy over time graph of an egress queue is depicted. In addition to the MAX, XOFF, and XON thresholds, the WARN threshold is added. As in the ingress queues, XOFF and XON thresholds are used to detect congestion and the lack thereof. MAX threshold is the absolute capacity of the buffer. If it is reached, the new packets are dropped. Before explaining the WARN threshold, we introduce a new variable $C_p^{i,o}$ which keeps the arrival count of the packets in egress queue of port o of priority p from input port i. These counters are used to identify the heaviest contributors to the congestion at the egress queues. However, they bring additional complexity since they need to be updated every time a new packet comes in to an egress queue. In addition, the freshness of these counters is also important to accurately identify the congestion contributors. Therefore, CaPFC introduces the WARN threshold to trigger those counters only in the congestion state. If the buffer occupancy is below the WARN threshold, the counters are reset to 0. When the buffer occupancy passes the WARN threshold, the counters are activated to keep the number of packets arriving from each ingress port for a specific egress queue. WARN threshold helps to reduce complexity and keep counters fresh and accurate. In Section III, we discuss how to tune these thresholds to optimize the performance.

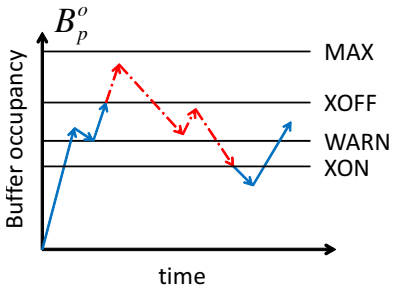


Fig. 4. Buffer occupancy over time for egress queue buffers.

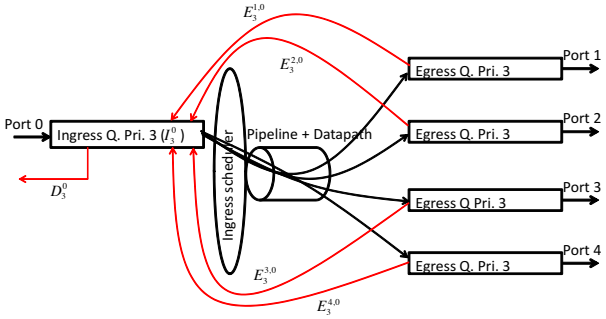


Fig. 5. Packets from a single ingress queue are forwarded to all egress queues per-priority. In return, the ingress queue receives a congestion state signal from each egress queue per-priority.

CaPFC introduces the variable $E_p^{i,o}$ to keep the congestion state of an egress port o for input port i and for priority queue p. These congestion state values are fed back to the corresponding input ports as shown in Fig. 5. These state variables are then used to determine to send XON or XOFF messages out of those input ports to the upstream devices. In CaPFC, egress queues do not pause the traffic coming out of ingress queues to avoid HOL which helps to preserve virtual parallel lanes for each priority.

Stop-Max is the name of the first algorithm that is used to calculate the congestion states where only the maximum contributor of the congestion is identified and the egress congestion signal $E_p^{i,o}$ is set to 1 only for that input port. If the congestion persists, the counter of other input ports continue to increase and the max contributor index changes. At that point, the new maximum counter input port is also put in congestion state and asked to pause upstream traffic. Input ports are put into congestion state one-by-one as long as congestion persists. The pseudocode of this algorithm is given in Algorithm 1.

The second algorithm is named Stop-Calibrate Policy. When the buffer occupancy is above the XOFF threshold, it

Algorithm 1 Stop-Max Algorithm

Require: $B_p^{o,+} = B_p^o$ + size of the new packet
Require: $B_p^{o,-} = B_p^o$ - size of the dequeued packet
1: **if** Enqueue a packet at port o with priority p **then**
2: **if** $B_p^{o,+} \geq MAX$ threshold **then**
3: Drop the packet
4: **else**
5: Accept the packet
6: $B_p^o \leftarrow B_p^{o,+}$
7: **if** $B_p^o \geq WARN$ threshold **then**
8: $C_p^{i,o} \leftarrow C_p^{i,o} + 1$
9: **if** $B_p^o > XOFF$ threshold **then**
10: Detect the heaviest contributor i^*
11: $i^* \leftarrow \arg\max_i C_p^{i,o}$
12: Put it into congestion state
13: $E_p^{i^*,o} \leftarrow 1$
14: **else if** Dequeue a packet from port o with priority p **then**
15: Remove the packet
16: $B_p^o \leftarrow B_p^{o,-}$
17: **if** $B_p^o \leq WARN$ threshold **then**
18: $C_p^{i,o} \leftarrow 0 \quad \forall i$
19: **if** $B_p^o \leq XON$ threshold **then**
20: $E_p^{i,o} \leftarrow 0 \quad \forall i$

first sorts the congestion counters of the egress queue. Then starting from the highest, it sets the congestion signal for the input ports whose cumulative counter percentages pass a predefined cut-off threshold ratio. This threshold can be calibrated between 0 and 1 defining the aggressiveness of the CaPFC technique. If the threshold ratio is set as 1, then congestion signal of every input port with a packet inside the egress queue is set to 1. In Algorithm 2, we only reflect the differences from the Algorithm 1. The steps 10 and 13 in Algorithm 1 are replaced by the steps in Algorithm 2. CUT is the calibrated cut-off threshold, the ratio of the congestion contribution that is determined to be heavy.

Algorithm 2 Changes for Stop-Calibrate Policy

1: Detect the heaviest contributors ▷ In place of Step 10
2: Sort congestion counters $C_p^{i,o}$ based on i
3: Find the set of input ports IP whose cumulative congestion contribution ratio passes the CUT threshold
4: $E_p^{i,o} \leftarrow 1 \quad \forall i \in IP$ ▷ In place of Step 13

E. Combined Congestion Reaction

In PFC, if the ingress queue of a certain priority gets into a congestion state, it sends a PFC-XOFF message to pause the incoming traffic. CaPFC incorporates both ingress queue state and egress queue states to pause and unpause the incoming traffic. An ingress queue may send packets to different egress queues of the same priority as depicted in Fig. 5. Likewise, it receives an egress state signal $E_p^{i,o}$ destined for this input port. If any of the egress ports sees the traffic coming from ingress queue i is responsible for the congestion at egress queue o, they will notify the ingress queue with the $E_p^{i,o}$ signal set to 1. Therefore, the decision to send an XOFF or XON message to the upstream device is based on the congestion state of the ingress queues and the congestion state signals received from all egress queues. In input port i, for priority p, the combined congestion state variable D_p^i is calculated by

$$D_p^i = I_p^i \vee \bigvee_{\substack{o \in P, \\ o \neq i}} E_p^{i,o} \qquad \forall i \in P, 1 \leq p \leq 8,$$

where P is the set of ports. For example, if we assume the switch has five input/output ports indexed as $0, 1, 2, 3, 4$, then the CaPFC decision variable for port 0 for priority 3 will be equal to

$$D_3^0 = I_3^0 \vee E_3^{0,1} \vee E_3^{0,2} \vee E_3^{0,3} \vee E_3^{0,4}.$$

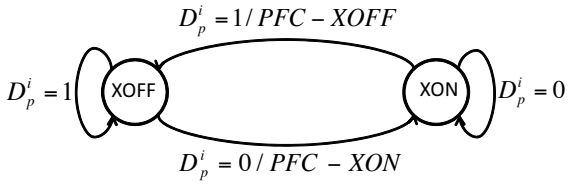


Fig. 6. The decision logic behind CaPFC, which is very similar to that of PFC except I_p^i is replaced by D_p^i.

In order to reduce complexity and overhead, we assume that congestion state signals E_p^i are only sent when there is a change of state. The ingress queue stores the latest updates of these signals and recalculates D_p^i only if at least one of the state signals change. The XOFF/XON behaviour of a receiving port based on variable D_p^i is highlighted with a finite state machine in Fig. 6. Compared to PFC, the only difference is the CaPFC messages are triggered by combined congestion state signals D_p^i instead of just ingress congestion state signals I_p^i. In short, the traffic incoming at port i with priority p is paused if the ingress queue is in congestion state or the ingress queue is deemed responsible for congestion at any of the egress queues, for that priority. The traffic for priority p is unpaused only if both the ingress queue is out of congestion state for that priority and is not responsible for the congestion in any of the egress queues for that priority.

F. Shared Memory Switches

In shared memory merchant silicon switches, every priority queue at every port has a small dedicated memory. The rest of the memory is assigned to a common pool. There are per-priority-queue, per-port, per-all-ingress-ports, per-all-egress-ports thresholds on the buffer usage from this shared memory pool. When admitting a packet from an ingress queue to an egress queue, the buffer occupancy counters of that egress queue, counters of that egress port and the total buffer occupancy of all egress ports are checked. If any of those counters are equal to the capacity threshold, this packet cannot be admitted. When an ingress queue admits a packet, it also checks the buffer occupancy counter of total shared memory usage in that switch. If there is no space left in the shared memory pool, the packet is not admitted. In our CaPFC implementation, setting XOFF, XON, and WARN thresholds for each buffer usage counter would be complex. Therefore, we implement CaPFC only on per-priority-queue buffer usage and set those thresholds with more safety margin to mitigate the buffer overflow with respect to other counters.

G. CaPFC Implementation Requirements

The goal behind CaPFC is to design a congestion aware flow control scheme with maximum added functionality but with minimum extra complexity. It is crucial that CaPFC is easily integrated to the legacy hardware in which PFC has already been deployed. As depicted in Fig. 7, CaPFC assumes no change on top PFC from a network perspective. The transmitter and receiver mechanisms of PFC is intact whereas the decision logic behind XOFF and XON messages is modified. Therefore, CaPFC does not require any change in the network interface cards (NICs) but requires monitoring of the egress queues, signaling between the ingress and egress queues and a modified logic in the ingress queues. All of these changes are easy to adapt in practical switches.

CaPFC introduces per-input-port per-priority counters at egress ports. CaPFC is envisioned to be used only for lossless traffic services. In a network, typically few of eight classes of services will use lossless services. Therefore, the counters for this purpose will be typically lower end of $O(P^2)$, where P is the number of ports. In popular commodity switches with 48 ports, that corresponds to maximum 18432 counters for 8 priorities, which is manageable in those switches. If each counter occupies 2 bytes of memory, it corresponds to approximately 36 KB extra memory, which is easily affordable. In high-radix switches with 128 ports, the CaPFC service would be enabled for a small number of lossless priorities, which keeps the number of total counters around 10K's.

The computational complexity brought by CaPFC is updating the congestion counters every time a new packet comes in and goes out of an egress queue and calculating the heaviest ingress ports. This complexity is significantly simplified after the introduction of the WARN threshold since only the lossless queues which observe high levels of buffer occupancies need to update their counters at the line rate. A software solution may not scale in the cases when multiple queues observe high occupancy levels, therefore a hardware solution is a better fit. A dedicated FPGA can keep the counters in a table per egress port and update them in line rate if the corresponding queue occupancy is over the WARN threshold. Once the occupancy of an egress queue passes the XOFF threshold, the FGPA module is responsible to sort the per-ingress-port counters, which has $O(PlogP)$ complexity. In Stop-Max algorithm, the congestion signal of the index of the maximum counter is set to 1. In Stop-Calibrate algorithm, the minimum set of the indexes whose counter contributions exceed the CUT threshold are found. It is carried out by cumulating the counters in the decreasing order until the threshold ratio is passed. It has a $O(P)$ complexity. Overall, CaPFC has a limited memory requirement and requires a small die space with $O(PlogP)$ computational complexity in an FPGA.

On the other hand, CaPFC mitigates the disadvantages of QCN. Unlike QCN, CaPFC can cross VLAN borders since it uses the same network interface of PFC as shown in Fig. 7. Second, it incurs much less complexity compared to QCN. In Stop-Max algorithm, it introduces only the WARN threshold. For simplicity, the WARN threshold is a virtual register which is represented by the XON threshold in the switch ASIC. In Stop-Calibrate algorithm, the cut-off parameter is also introduced. On the other hand, a typical QCN implementation introduces 12 extra parameters [7]. The tuning complexity of those parameters are also much higher. In addition, CaPFC works per-input port whereas QCN works per-flow, which also increases the complexity of the operation. Finally, unlike QCN, CaPFC does not require any change in the end-hosts. In conclusion, CaPFC adds significant congestion control functionality to flow control without introducing compatibility issues and significant complexity concerns.

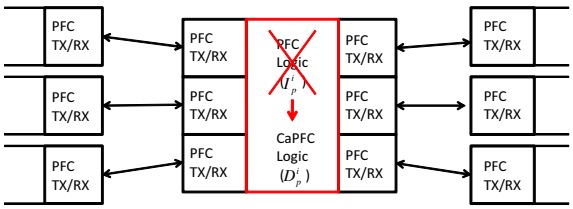


Fig. 7. CaPFC does not change the transceiver structure already deployed for PFC. CaPFC only replaces the decision logic of PFC by incorporating egress queue states along with the ingress queue state per-priority per-input port. The ingress decision variable I_p^i of PFC is replaced by combined decision variable $D_p^i = I_p^i \vee \bigvee_{o \in P,\ o \neq i} E_p^{i,o}$ of CaPFC.

H. Cross Layer Interaction

We believe a comprehensive solution should come from a multi-layer approach which integrates different techniques. Therefore, we built CaPFC on the cross-layer architecture of DeTail [8]. In DeTail, besides a lossless fabric and hierarchical prioritization, there are changes in the network layer and transport layer. In network layer, it proposes a per-packet adaptive load balancing scheme which dominates Flow Hashing and Lossless Packet Scatter in terms of reducing congestion hotspots [8]. In addition, the transport layer is modified to be reorder-resistant since packets can take different routes due to Adaptive Load Balancing. The transport layer of DeTail uses explicit congestion notification (ECN) protocol to detect congestion. CaPFC interacts with TCP in the same way PFC does as explained in [8]. CaPFC redesigns the link layer portion of DeTail but requires no change in the network and transport layers except tuning of the parameters.

III. Parameter Selection Analysis

Setting parameters in PFC is limited to finding XOFF and XON thresholds assuming the MAX threshold is given by the hardware specs. It is clear that XON must be smaller than XOFF. In selecting the XOFF threshold, the idea is to put the minimum amount of headroom buffer that will absorb all of the traffic that was received after a PFC pause message is sent to the upstream device [9]. To calculate this value, maximum transmission unit (MTU) on the transmitting end of receiver, length of the cable, speed of the wire, transceiver latency, response time of sender, and MTU on the transmitting end of sender are taken into account. To simplfiy, the headroom buffer must be approximately equal to maximum link capacity times RTT between two links. The XOFF threshold is set by subtracting the headroom buffer from the MAX threshold. Setting XON threshold is more challenging than the setting the XOFF threshold. Setting it too low causes buffer underflow, low link utilization and congestion spreading [8]. Setting it too high causes too many PFC messages which creates too much overhead in return [8]. We tuned the value of XON threshold according to the on simulation results.

In CaPFC, XOFF and XON thresholds of the ingress buffers are set the same as in PFC. At the egress buffers, setting XOFF threshold requires a deeper analysis. Even if the egress queue sends congestion state signals to trigger a pause message at all input ports, it needs to take the packets already in the ingress buffers and on the links into account. However, taking the worst case into account may end up with an extremely large headroom buffer which makes CaPFC unfeasible. Therefore, a statistical multiplexing approach is adopted since egress buffer overflow is not desirable but tolerable. The cost of egress overflow with very low probability is usually less than under utilization of buffer and link capacities. We tuned to the optimal value by simulations. XON threshold in egress buffers is set with the same idea in the ingress buffers.

The WARN threshold is set less than the XOFF threshold to have a fresh picture of the congestion contribution before asking the input ports to send pause messages. However, it should not be set less than the XON threshold because in persistent congestion states, the buffer occupancy may not drop well below the XON threshold. Therefore, the counters may be not be reset for a long time if the WARN threshold is set lower than the XON threshold. It results in expired statistics about the contributors of congestion. We set the WARN threshold the same value as the XON threshold at the egress buffers because it enables using the existing XON threshold in hardware for the purposes of the WARN threshold to simplify the implementation.

IV. Performance Evaluation

A. NS-3 Simulation Setup

The NS-3 simulation setup is built using the code base of DeTail [8] work to evaluate the performance of CaPFC compared to PFC. Even though it is missing in [8], the code base uses Network Simulation Cradle (NSC), a framework that embeds real Linux code in the NS-3 simulation. It enables to get more realistic results and it has ECN functionality. We also adopted the topology based on the input from Section 5.4 of [10] which is larger than the one in [8]. The simulated switch has a pipeline based packet processor as presented in Section II-A. It employs FIFO and ETS for ingress and egress scheduling, respectively. The parameters of this switch are taken from [8] which assumes 25μs switch delay. We vary the packet processing speed of the pipeline-based switch from $1M$ packets/s to $0.5M$ packets/s. ECN is implemented using NSC's TCP stack. ECN threshold is set to $20KB$ as optimized by the simulations. The adaptive load balancing thresholds explained in [8] are set to $16KB$ and $32KB$, respectively.

B. Traffic Characterization

In data center networks, traffic can be characterized by mainly three different traffic types namely real-time query traffic, latency-sensitive short messaging traffic and throughput-oriented long background traffic [3]. Query traffic is also sensitive to latency in addition to having a risk of throughput collapse named as TCP Incast [11]. We simulate all three traffic patterns concurrently inside the network. Inter-arrival times of the long traffic and short traffic flows are exponentially distributed with a mean of 5000 and end-nodes of the flows are uniformly selected. Query traffic consists of randomly selected 40 source nodes sending a fixed size of flow to a randomly selected common aggregation node simultaneously. The inter-arrival time of the query traffic is also exponentially distributed with the expected arrival rate of 100. These traffic types are differentiated from each other by setting different priorities. Some of these priorities may use Strict Priority (SP) Queues, whereas some others can go over Weighted Deficit Round Robin (WDDR) scheduling. We created six traffic simulation scenarios for the dedicated buffer memory and one for shared buffer memory setup to test the proposed techniques in different conditions. The traffic scenarios for the dedicated memory setup are given in Table I. PS is the packet processing speed of the pipeline in terms of packets per second.

TABLE I. Different Traffic Simulation Scenarios

		Long Traffic		Short Traffic		Query Traffic	
Scenario No	PS	Size	Queue Type	Size	Queue Type	Size	Queue Type
1	1Mpps	1MB	WDRR	64KB	SP	16KB	WDRR
2	1Mpps	1MB	WDRR	16KB	SP	16KB	WDRR
3	1Mpps	1MB	WDRR	64KB	SP	32KB	WDRR
4	1Mpps	1MB	WDRR	64KB	WDRR	16KB	SP
5	1Mpps	1MB	WDRR	64KB	WDRR	16KB	WDRR
6	0.5Mpps	1MB	WDRR	64KB	SP	16KB	WDRR

C. Simulation Results

In this section, we investigate the performance of two versions of CaPFC compared to two versions of PFC used in DeTail in terms of tail (maximum) flow completion time (FCT). CaPFC_max implements the Stop-Max algorithm whereas CaPFC_cal implements the Stop-Calibrate algorithm. In PFC with drop (PFC_wdrop), a packet is dropped at the end of pipeline if the destination egress queue has no space, whereas PFC with stop (PFC_wstop) pauses packet processing in that case to prevent any packet drop.

The test network is a Fat-Tree topology with 8-port 128 servers and 80 switches from [10]. Since the simulation employs NSC, it is not easy to try larger networks. The capacity of the links are $1Gbs$ and the packet processing speed is either $1M$ packets/s or $0.5M$ packets/s depending on the simulation scenario in Table I. Total subscription ratio is 4, two from ToR to aggregate and two from aggregate to core. The values of parameters are given in Table II. We note that MAX, XOFF, and XON thresholds are per-port per-priority.

TABLE II. Simulation Parameters

Parameter	Value	Parameter	Value
Pipeline speed	0.5Mpps or 1Mpps	Ingress XON thres.	40KB
Switch delay	25μs	Egress MAX thres.	60KB
ECN thresh.	20KB	Egress XOFF thres.	25KB
ALB thresh. (min/max)	16KB/32KB	Egress XON thres.	20KB
Ingress MAX thresh.	60KB	Egress WARN thres.	20KB
Ingress XOFF thres.	50KB	CUT thresh.	80%

We take the simulation scenario 1 as the base case and observe the effect of different parameters on the performance by changing them one-by-one in each scenario. In the base scenario, short message traffic has higher priority than the rest of the traffic. In Fig. 8(a), tail FCT of four different techniques are shown with a breakdown in terms of traffic type. The major observations are detailed in the following sections.

1) CaPFC provides high burst absorbtion for query traffic: Both versions of CaPFC reduces the maximum FCT of the query traffic approximately 6 times and 3 times compared to PFC_wdrop and PFC_wstop, respectively. CaPFC absorbed the traffic bursts in a much better way than both versions of PFC. Query traffic experiences TCP incast most severly

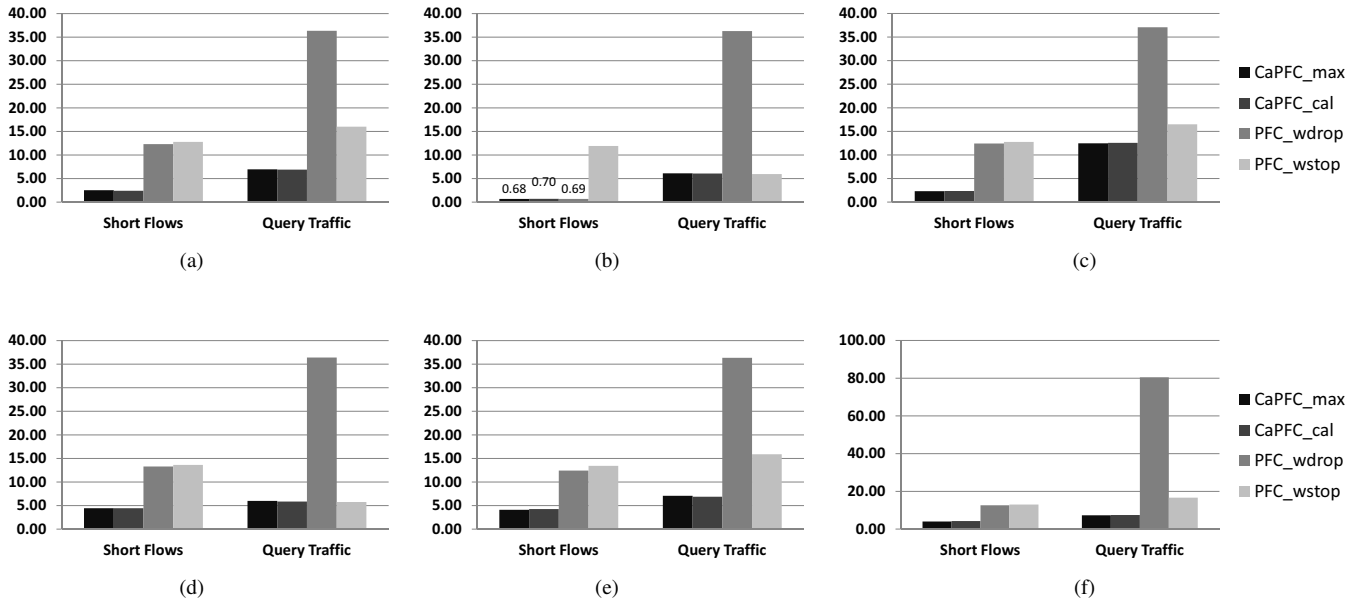


Fig. 8. Comparison of different versions of CaPFC and PFC in terms of tail FCT (unit of y-axis is miliseconds) in (a) simulation scenario 1, (b) simulation scenario 2, (c) simulation scenario 3 (d) simulation scenario 4, (e) simulation scenario 5, and (f) simulation scenario 6.

under PFC_wdrop mechanism since sudden bursts lead to synchronous packet drops.

2) CaPFC satisfies low latency for the short message flows: In terms of tail latency of the short message flows, both versions of CaPFC resulted in approximately 5 fold savings compared to both versions of PFC. PFC_wdrop suffers from fast retransmissions and timeouts whereas PFC_wstop suffers from HOL blocking and congestion spread.

In the rest of the evaluation, we change some of the simulation and traffic parameters one-by-one to see their effect on the performance of techniques. These scenarios are outlined in Table I from scenario 2 to 6.

3) CaPFC is efficient in different flow sizes: In scenario 2, the size of the short flows are decreased from $64KB$ to $16KB$. Similarly, in scenario 3, the size of the query traffic flows are doubled to $32KB$ compared to the base case. In Fig. 8(b) and Fig. 8(c), the tail FCT of the query and the short messaging traffic are shown for simulation scenario 2 and 3. Interestingly, in Fig. 8(b), PFC_wdrop incurs approximately 17 times more tail FCT for short flows whereas PFC_wstop incurs approximately 6 times more tail FCT for the query traffic. The results in Fig. 8(b) and Fig. 8(c) are consistent with the results in Fig. 8(a), which means both versions of CaPFC performs good for different flow sizes.

4) CaPFC is efficient in different prioritization configurations: In scenarios 4 and 5, we change the queue types of the short and query flows. Originally, short message traffic has the highest priority. In simulation scenario 4, the query traffic has the highest priority whereas in scenario 5, all traffic types go through separate WDRR queues. The results are presented in Fig. 8(d) and Fig. 8(e). These results are also consistent with the results in scenario 1.

5) CaPFC is efficient in different packet processing speeds: Lastly, we decrease the processing speed of the pipeline to half. It puts the pressure more on the ingress buffers therefore egress buffer occupancy becomes less important. According to the results in Fig. 8(f), changing the speed of the pipeline does not curb the advantage of CaPFC over PFC_wstop but amplifies over PFC_wdrop.

TABLE III. LONG BACKGROUND TRAFFIC TAIL FCT

	Tail FCT (ms) Based on Scenario					
Technique	Sce. 1	Sce. 2	Sce. 3	Sce. 4	Sce. 5	Sce. 6
CaPFC_max	52.56	51.08	55.13	58.55	50.87	58.32
CaPFC_cal	53.21	55.34	51.77	57.80	49.32	55.55
PFC_wdrop	50.67	49.80	62.42	56.78	51.57	56.08
PFC_wstop	55.22	55.87	52.17	53.46	61.02	59.75

6) CaPFC offers high throughput for the long background flows: The tail FCT of the long background flows are given in Table III for every scenario. Even though CaPFC resulted in significant savings in terms of tail latencies of the query traffic and short message traffic, it did not compromise the bandwidth requirement of the long background flows. Both versions of CaPFC had similar results compared to both versions of PFC. In the worst case scenario, CaPFC results in 4% increase in the tail FCT compared to PFC_wdrop in Scenario 6. It improves the tail FCT up to 11.7% and 16.6% compared to PFC_wdrop and PFC_wstop, respectively.

7) CaPFC is efficient in shared memory architectures: We modify the switch architecture to observe the performance of both versions of CaPFC in shared memory switches. Each priority queue has $8KB$ dedicated memory and there is total $350KB$ available memory in the shared pool. The maximum shared buffer capacity that can be used by each priority queue and by each port are $80KB$ and $100KB$, respectively. In the egress queues, we set the XOFF, XON, and WARN thresholds to $50KB$, $35KB$, and $35KB$, respectively. In the ingress queues, XOFF threshold is set to $30KB$ whereas XON threshold is set to $20KB$. The traffic scenario is given as scenario 1

in Table I. The tail FCT of query traffic and short messaging traffic are given in Fig. 9. Even though both versions of CaPFC perform significantly better than both versions of PFC for the query traffic, CaPFC_cal is 3 times better then CaPFC_max. Shared memory architecture helps both versions of PFC in terms of tail FCT of the short messaging traffic. However, PFC_wdrop doubles the tail FCT of short messaging traffic compared to both versions of CaPFC and PFC_wdrop.

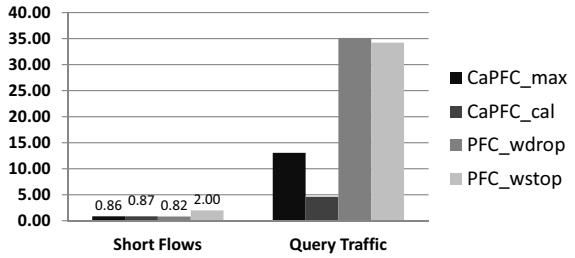


Fig. 9. Tail FCT (ms) results for all techniques in a shared memory switch.

V. Related Work

Flow control and congestion control are addressed in different layers and in different places in the network. Recently, limitations of PFC, such as HOL blocking and unfairness, are highlighted in [12]. Authors propose DCQCN a new congestion protocol based on QCN and DCTCP. DCQCN is an end-to-end protocol that involves higher layers. As an alternative, we propose a purely link-layer-based hop-by-hop solution. In [13], limitations of PFC are also listed as large buffering delays, unfairness, HOL blocking, and deadlock. Similarly, PFC is combined with DCTCP [3] and a deadlock-free routing scheme is proposed in [13]. In this paper, we assume no changes to TCP and propose a solution based on the link layer, which can react faster to flash congestion. A different approach is taken in [14], a modified QCN technique is proposed to reduce the side effects of PFC. It complements per-port per-priority granularity of PFC with the per-flow granularity of QCN. However, it inherits all three limitations of QCN described in Section I.

In [7], the impacts of PFC and QCN over different TCP mechanisms are investigated. The results suggest that PFC improves the TCP performance in every case and emphasize on the deficiencies of QCN as burst sensitivity, lack of adaptivity and unfairness. The performance of PFC over different traffic scenarios is investigated in [15], which elaborates the limitations of PFC, which are starvation and unfairness, arising due to interactions of different flows. Compared to these works, CaPFC improves the benefits of PFC by alleviating some of these limitations utilizing from both ingress and egress queue statistics to take proactive action earlier.

Infiniband uses a credit based flow control algorithm to ensure lossless service [16]. A receiver advertises to the sender the free capacity in its buffers and the sender does not exceed that limit while transmitting data. However, Infiband is much less ubiquitous than Ethernet. Virtual output queueing (VOQ) is also used to alleviate HOL blocking when there is an association between ingress and egress queues, which is not possible at the switch architecture we take reference in Section II-A. VOQ is a costly operation but more importantly, it does not differentiate between heavy and light flows. Therefore, it leads to long queueing delays for light short flows, when the egress queue is full.

Congestion management has been a popular topic in data center networks for some time. Most of the recently proposed techniques address this problem at the network or transport layer. The survey in [17] breaks down the techniques dealing with this problem into four subcategories, reducing queue length, accelerating retransmission, prioritizing mice flows, and exploiting multipath. DCTCP [3] is acknowledged to be a seminal work in this field. DeTail [8] is one of those techniques and is unique for utilizing both prioritization and multipath. It is also unique because it implements PFC to provide lossless link layer operation to complement the per-packet adaptive load balancing algorithm that reduces tail flow completion times in data center networks. However, in [8], the main switching mechanism is characterized as a crossbar rather than a pipeline mechanism, which is taken as the main reference in this paper. Also in [8], analysis about different switch types is shallow.

Finally, the lack of fairness in QCN is pointed out by [11] and [18] and its effect on the network performance for the TCP incast scenarios are shown and a modified version of QCN is proposed, namely FQCN in [11], to mitigate this drawback. [18] also proposes a modification to QCN, namely AF-QCN, which provides faster fairness convergence and enables weighted fairness. Even though these modifications improve the performance of QCN, they could not help it to gain widespread adoption.

VI. Conclusion

PFC and QCN are proposed as parts of DCB to enhance the capabilities of Ethernet. However, they have certain limitations that bars their wide deployment. First, PFC only relies on monitoring ingress queues, therefore, it is not protected against saturation trees and HOL. Second, QCN has multiple issues but most importantly it cannot pass VLAN borders. As a remedy to the shortcomings of PFC and QCN, we propose CaPFC as a congestion aware flow control mechanism. CaPFC monitors both ingress and egress queues and takes proactive action to prevent egress buffer overflow which causes congestion spread and HOL blocking.

By implementing CaPFC, we achieve four features of low latency data center networks. First, we use prioritization and use SP queueing for latency-sensitive flows. Also, we provide parallel lanes for each type of traffic by preventing congestion in one type of traffic to spread to other types of flows. Second, by proactively monitoring both ingress and egress queues, we minimize HOL blocking and retransmissions due to packet drops. This in turn increases link utilization and reduces average queue length, which also reduces latency.

We carried out simulations in NS-3 to evaluate the performance of CaPFC compared to PFC. The results show that CaPFC is able to reduce maximum FCT of the query traffic up to 6 times. It also reduces the maximum FCT of the short messaging traffic up to 17 times. In addition, CaPFC keeps the performance improvement on various traffic scenarios and switch parameters. On the other hand, it does not sacrifice the bandwidth requirement of the long background traffic at

the expense of dramatic performance of the latency-sensitive traffic. For future work, we plan to implement CaPFC with different transport protocols, such as DCTCP, and different load balancing schemes, such as [10].

As a future work, we also consider to test CaPFC in a realistic topology with real network loads to observe tighter queue limits as the number of ports in the switch and the number of senders in query traffic increase. We expect that CaPFC would still provide low latency for short and query traffic flows at the cost of little bandwidth loss for large flows.

REFERENCES

[1] K. Won, "Trends reshaping networks," http://www.networkworld.com/article/2198766/tech-primers/trends-reshaping-networks.html, 2011.

[2] D. Crisan, "Optimized protocol stack for virtualized converged enhanced ethernet," Ph.D. dissertation, ETH Zurich, 2014.

[3] M. Alizadeh et. al., "Data center TCP (DCTCP)," in *Proc. SIGCOMM '10*, vol. 1, New York, 2010, pp. 63–74.

[4] "Quantized congestion notification and todays fibre channel over ethernet networks," Cisco Systems, Inc, Cisco Website, Tech. Rep., 2014.

[5] "High capacity StrataXGS® Trident II ethernet switch series," http://www.broadcom.com/products/Switching/Data-Center/BCM56850-Series, 2014.

[6] M. Gusat, C. Minkenberg, and G. J. Paljack, "Flow and congestion control for datacenter networks," IBM, Tech. Rep., 2009.

[7] D. Crisan et. al., "Short and fat: Tcp performance in CEE datacenter networks," in *Proc. HOTI 2011*, Santa Clara, CA, August 2011.

[8] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: reducing the flow completion time tail in datacenter networks," in *Proc. SIGCOMM '12*, vol. 1, New Delhi, 2012, pp. 139–150.

[9] "Priority flow control: Build reliable layer 2 infrastructure," Cisco Systems, Inc, Cisco Website, Tech. Rep., 2009.

[10] J. Cao et. al., "Per-packet load-balanced, low-latency routing for clos-based data center networks," in *Proc. CoNEXT '13*, Santa Barbara, CA, December 2013.

[11] Y. Zhang and N. Ansari, "On mitigating TCP incast in data center networks," in *Proc. INFOCOM 2011*, Shangai, April 2011.

[12] Y. Z. et. al., "Congestion control for large-scale RDMA deployments," in *Proc. SIGCOMM '15*, vol. 1, London, 2015, pp. 523–536.

[13] B. Stephens et. al., "Practical DCB for improved data center networks," in *Proc. IEEE INFOCOM 2014*, Toronto, April 2014, pp. 1824–1832.

[14] F. D. Neeser et. al., "Occupancy sampling for terabit CEE switches," in *Proc. HOTI 2012*, Santa Clara, CA, August 2012.

[15] M. Haggen and R. Zarick, "Performance evaluation of DCBs priority-based flow control," in *Proc.* 10^{th} *International Symposium on Network Computing Applications (NCA)*, Cambridge, MA, August 2011.

[16] "Infiniband FAQ, rev 1.3," Mellanox Technologies, Mellanox Website, Tech. Rep., December 2014.

[17] S. Liu, H. Xu, and Z. Cai, "Low latency datacenter networking: A short survey," *http://arxiv.org/abs/1312.3455*, 2014.

[18] A. Kabbani et. al., "Approximate fairness with quantized congestion notification for multi-tenanted data centers," in *Proc. HOTI 2010*, Mountain View, CA, August 2010.

Overlaying Delay-Tolerant Service using SDN

Patrick Maillé
Telecom Bretagne
Rennes, France
patrick.maille@telecom-bretagne.eu

Shyam Parekh
AT&T Labs
San Ramon, USA
shyam.parekh@att.com

Jean Walrand
University of California
Berkeley, USA
walrand@berkeley.edu

***Abstract*—Telecommunication networks are generally dimensioned to provide services with small delays and high throughput during peak-periods. Due to the sizable difference in the network utilization between the peak and off-peak periods as well as the requirements of robust performance in face of both traffic burstiness and various types of network failures, these networks are significantly over-dimensioned for the average network loads.**

In this paper, we propose to use this extra capacity for supporting a deferrable traffic class with some guarantees on its end-to-end delays. Using the Software-Defined Networking (SDN) capabilities for controlling the network ingress rates of the deferrable traffic class in real time, we ensure that such a service would remain transparent to existing delay-sensitive traffic. To estimate the available capacities for the deferrable service, we analyze large deviations for the proposed traffic model.

Starting from an initial network designed for delay-sensitive traffic, one can readily "overlay" a new network for the deferrable service at no extra cost. This overlaid network has the same topology as the original one, and its link capacities can be directly computed from the characteristics of the existing traffic, the original link capacities, and the end-to-end delay tolerances.

I. Introduction

Telecommunication networks have been witnessing an exponential increase in traffic volumes since the 1990s, driven in the last years by the widespread adoption of cloud services, the generalization of 4G mobile usage, and the user consumption changes from television to video streaming. This trend is very likely to continue with the advent of 5G and the higher definition of videos viewed online, putting again more pressure on the infrastructure owners to increase transmission capacities.

A comparable increase in demand is observed in electric power distribution networks. There, many solutions are envisioned to reduce infrastructure, production, and/or environment costs by smoothing out the (also highly variable) demand. Those solutions include *deferring* part of the demand in exchange for a lower unit price, and quantitative analyses show how much can be saved, for example when different types of demand have different deadlines [4], [5].

Telecommunication network features differ from electric distribution, including faster variations over time and the absence of alternative sources of supply. (Note that [4], [5] focus on the use of renewable energy, but assume that grid power is always available.) Nevertheless, we believe the idea of deferred traffic–treatable within a deadline with high probability–is worth investigating also there. A typical example is for video on demand: consumers could be asked to select a movie *in advance*, which would then be "pushed" through the network within a deadline, using only the capacity left unused by other flows instead of being downloaded or streamed as a delay-sensitive flow. Such a new service would then be transparent to existing traffic, and could help postpone capacity investments through a more efficient use of the existing infrastructure.

In this paper, we use large deviations analysis [16], [18] to estimate the amount of capacity that could be used by deferrable traffic. The idea is to control the probability that the average capacity available over some duration T is insufficient to carry some amount of deferrable traffic: given T and a target failure probability, we compute a corresponding capacity for deferrable traffic. While the analysis only provides results for the *rate* at which that probability decreases with T, simulation results show that ignoring smaller-order terms leads to very good estimates of the available capacity for deferrable traffic.

In terms of implementation, our approach relies on the current Software-Defined Networking (SDN) efforts [1], [11], in that it can leverage the use of logically centralized controllers, aware of the current network conditions, to inject deferrable traffic so as to remain transparent to delay-sensitive flows. Beyond the controller, some other management tools can also be applied, ranging from lower prioritization of deferrable traffic to more elaborate methods aimed at reducing the need for buffering in intermediate nodes, such as the Fastpass approach proposed in [15]. Since deferrable traffic will use the volatile resource left available by non-deferrable flows, we can also imagine that the routing applied to deferrable traffic be subject to rapid changes in order to optimize the instantaneous throughputs; this again implies the knowledge of the current network states, and the capacity to impact rapidly the behavior of routers through interfaces such as OpenFlow [13].

We are not the first ones to apply large deviations to analyze delays. In [17] the focus is on scheduling jobs in a multi-class queue so that out-of-time probabilities decrease at target rates for each class when the tolerated delay increases. Considering the network aspect, the large deviations of a network of G/G/1 (single-class) queues are analyzed in [3], also at the job (or packet) level. With regard to those references, our interpretation of delays here is for fluid-like models (continuous flows), not jobs. Additionally, our methodology focuses on estimating the throughput that can be offered to a low-priority service for a given tolerated delay, while the references focus on the performance for high-priority jobs: in [17] the objective is to minimize out-of-time probabilities, and [3] analyzes the waiting times and queue lengths (two notions we do not have in this paper for nondeferrable traffic given our "session" modeling for non-deferrable flows).
Large deviations are also applied, again in the power grid context, in [14] to control the risks of delaying some part of

ISBN 978-3-901882-84-5, IFIP Networking 2016

the energy demand from specific devices (pool pumps) with specific constraints (e.g., at least one cycle per device per 24 hours). In this paper we consider a steady-state setting of demand (e.g., during the peak hour) instead of relying on partially predictable (daily) cycles in demand, and we concentrate on providing some deferrable service at a constant *perceived* rate, the delay being a consequence of variations in the primary use of the network.

Our approach is close to *stochastic network calculus* [6], [7]; the main differences being twofold. First, most stochastic network calculus models consider random arrival flows served by a (non-random) network node, while here the service provided is the capacity left unused by nondeferrable traffic, hence randomness on the *service* side. Second, while the goal in network calculus is to provide conservative bounds (e.g., on usable capacity for delay to be below a threshold with high probability), we intend here to estimate the actual value, and for that we treat the large-deviation results (giving the speed for deviation probabilities) as "direct" estimates. Extensive simulations highlight the accuracy of this method. In stochastic network calculus, the closest notion to what we are investigating is that of *leftover capacity*, studied in [2], where the focus is still on obtaining bounds rather than on approaching the actual value. The contribution of this paper is then a method to estimate the usable capacity for given quality constraints given the characteristics of the nondeferrable traffic using it, and its extension in a very simple manner to the network case: it is indeed sufficient to apply the single-link method independently on each link of a network.

Our work is also related to the literature on delay-tolerant networks [8], [19], but the paradigm is sensibly different. Indeed, delay-tolerant networks are generally studied in a wireless context, the changes in connectivity coming from node mobility, hence a focus on routing [20] and buffering [12] strategies. In contrast, here the topology is assumed fixed and the instantaneous "connectivity" (the available capacity) results from demand variations over time of the non-deferrable service, which can be studied with a specific stochastic model. Studying that stochastic model to infer delay guarantees for the deferrable traffic is the main focus of this paper.

For any tolerable delay T, our method provides an estimate for the available capacities on a global network, obtained from a per-link analysis. The outcome of the analysis is a possibly simple exploitation of those unused resources in the near future, through the coordination possibilities offered by the SDN paradigm. Numerical examples show that even for networks optimized for delay-sensitive traffic, capacity utilization can be raised to 95% by adding deferrable traffic, while in current practice it is limited to at most 75-80%, and quite often in the vicinity of 50% due to the time-of-day and day-of-year traffic variations as well as inherent traffic burstiness and the provision of backup paths to be used in the event of failures. Hence, we think our proposition has the potential to enable new types of services without incurring any cost for additional capacity.

The remainder of the paper is organized as follows. The general model considered in the paper is presented in Section II, while Section III treats the special case of one communication link. A simple network case is detailed in Section IV, highlighting the key difficulties in the extension to more complex topologies, in particular insisting on the necessity of caching deferrable traffic in intermediate nodes. Section V summarizes the implications of our results for the "deferrable service network" that can be defined on an existing network, by explaining how to estimate the capacity of each link of this overlay network to satisfy delay constraints while remaining transparent to the non-deferrable traffic. Conclusions and directions for future work are given in Section VI.

II. General model

We consider a peak period during which the non-deferrable traffic is assumed in steady-state. We focus on links that carry the traffic of many users, such as backbone links (as schematized in Figure 1) and possibly backhaul links. Those links are now facing congestion issues because of the demand increase but also because of the increase in last-mile capacities. We nevertheless assume that access capacities are still the bottleneck for users most of the time, i.e., the network is designed so that users use all of their access capacities when active. We use the term *sessions* to refer to user flows, assumed with a constant throughput equal to their access link capacity. Thus, we consider that the network is dimensioned to offer a throughput limited only by the access rate, with a high probability. In that sense, we neglect sessions (flows) that are too short to reach the access transmission rate. A way to include those "mice" in our model is to average, for each given link, their throughputs and to subtract them from the link capacity. This corresponds to assuming that those sessions are such that their aggregate rate is approximately constant at the scale of the acceptable delays for deferrable services.

We also assume that all users have the same access rate, denoted by b. Therefore, when a number X of users have their sessions use a given link (considering fixed routing per flow), the used capacity is simply Xb. If the link has capacity $\underline{C}$, there consequently remains some bandwidth $\underline{C} - Xb$ that can be used for our new (deferrable) service. In the rest of the paper, b will be taken as the capacity unit.

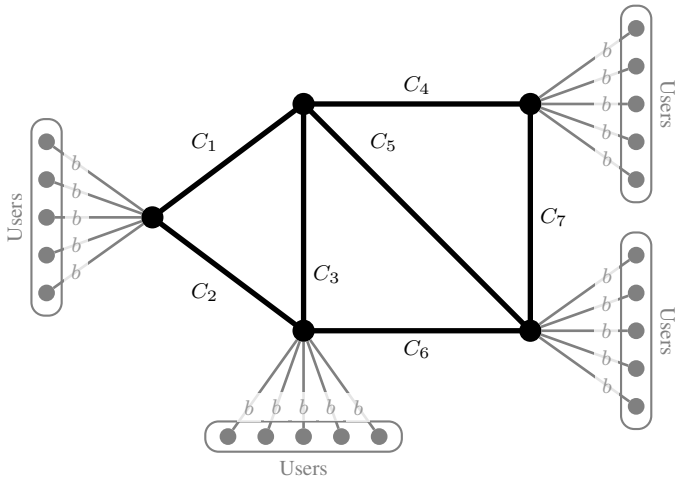


Figure 1. The type of backbone network considered (in black): individual rates are limited by user access rates. Grey parts schematize the access network, nodes in black can be entry points (for users and/or content providers) or simply intermediate nodes. All links are labeled by their capacity.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

We consider large numbers of users connected to each entry point, that behave independently. As a consequence, we assume that user sessions arrive according to independent Poisson processes. We moreover model session durations as exponentially distributed random variables with a common average duration denoted by $1/\mu$. Finally, we assume session routing is fixed, as least statistically: for each session route, arrivals follow a Poisson process. In practice, routes may adapt to network conditions, but since we consider networks that are dimensioned to keep saturation rare, we ignore that effect.

The question we now ask regards the use of the remaining capacity for deferrable service: given backbone link capacities, session arrival rates and average duration, we intend to offer a service based on that capacity, with looser delay constraints. More specifically, we want to choose a deadline T and offer a service for which delay is guaranteed to be below T with some high probability. In this paper we show how to compute the amount of such deferrable traffic that can be carried by the network, as a function of T, of the network capacities, and of the non-deferrable traffic characteristics.

We are aware that some of the assumptions we make are a considerable simplification of reality, but we believe the model we build on them provides useful insights regarding the potential offered by resources temporarily left idle by non-deferrable traffic.

III. The case of one link

In this section, we consider the case of a single (backbone) link, and detail the reasoning that will be applied in later sections to more complex topologies.

A. Setting and mathematical formulation

We denote the request arrival rate–assumed constant over the considered period–by λ (arrivals per time unit). Each session uses the same bandwidth b due to last-mile capacity limits, and goes through a single backbone link with capacity $\underline{C}$. As stated in Section II, the service duration of each request is assumed to follow an exponential distribution with parameter μ, so that if we assume that requests arriving while the link is full are rejected, the process describing the evolution of the number of active requests over time is an M/M/C/C queue [10], with $C := \lfloor \underline{C}/b \rfloor$.

Then the blocking probability for a non-deferrable request is simply given by the Erlang-B formula $B(\rho, C) = \frac{\rho^C/C!}{\sum_{k=0}^{C} \rho^k/k!}$ with $\rho := \lambda/\mu$. This formula can be used either to dimension the link (decide the value of C) for a given demand level ρ, or to decide how many users to route through this link (decide the value of λ).

In practice, the requests arriving while the link is fully used may not be rejected but rather be re-routed, or have to share the link capacity with existing sessions (although we can imagine an admission control scheme actually rejecting those requests). But we assume the decisions (on C or λ) are such that this occurs with small probability, so that the M/M/C/C model would still be a good approximation.

We consider providing deferrable service at an effective throughput represented by D, the equivalent number of access links with capacity of b bit/s each. For example, $D = 3$ corresponds to a effective throughput of $3b$ bit/s. The question is: depending on T and on the target probability of delay remaining below T, what value of D can the network handle? Equivalently, for given D and T, what is the probability that the amount DT of deferrable traffic is carried before the deadline T? The network controller will limit the amount of deferrable traffic to a value for which that probability is acceptably large, say 99%.

To address the question, let us consider a deferrable bit that enters the network at time t. If the network provides a first-come-first-served service for deferrable demand, that bit will be served after all the deferrable bits that arrived in the time interval $[t-T, t)$, since the value of D is chosen so that the delay does not exceed T. The probability that our considered bit can be served before $t+T$ at least equals the probability that the capacity left unused by the non-deferrable traffic during $[t, t+T]$ exceeds DT, i.e.,

$$\mathbb{P}\left(\int_t^{t+T} (C - X_\tau)\mathrm{d}\tau \geq DT\right) = 1 - \mathbb{P}\left(\frac{1}{T}\int_t^{t+T} X_\tau \mathrm{d}\tau > C - D\right), \quad (1)$$

where X_τ is the number of active users of the non-deferrable service at time τ.

The situation is illustrated in Figure 2 for a given realization of non-deferrable traffic: the network can offer an equivalent throughput D to a delay-T deferrable traffic if the average idle capacity over a duration T exceeds D with a sufficiently high probability.

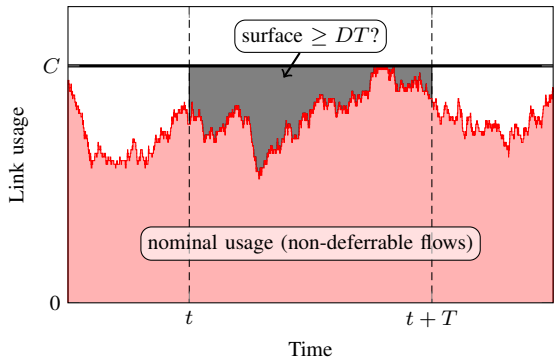


Figure 2. A trajectory for nominal (non-deferrable) usage, and the corresponding instantaneous available capacity (C=65, λ =50, μ=1, T=5).

In the following, we will therefore look for the relation between $A > 0$, T, and the "failure" probability

$$P_{A,T} := \mathbb{P}\left(\frac{1}{T}\int_0^T X_\tau \mathrm{d}\tau > A\right) \quad (2)$$

where (X_τ) is a continuous-time Markov chain corresponding to the number of clients in an M/M/C/C queue with offered load ρ, and X_0 is assumed to be distributed according to the stationary distribution of X, i.e., $\mathbb{P}(X_0 = x_0) = \frac{\rho^{x_0}/x_0!}{\sum_{k=0}^{C} \rho^k/k!}$ for $x_0 = 0, 1, \ldots, C$. We call $P_{A,T}$ the failure probability, since for $A = C - D$ it gives the probability that the average available capacity for deferrable traffic over T is below D, as indicated in (1).

B. Large deviations analysis

For delay durations that are large (e.g., with respect to the mean session duration $1/\mu$), the probability $P_{A,T}$ in (2) can be studied using large deviations [16], [18], and should then verify

$$P_{A,T} = e^{-TI(A)+o(T)} \tag{3}$$

where

$$I(A) := \sup_{\theta\in\mathbb{R}} \left[\theta A - \Lambda_\theta\right], \tag{4}$$

with Λ_θ the principal eigenvalue (eigenvalue with largest real part) of the matrix $Q+\theta V$, V a diagonal matrix with $V(i,i) = i$ for $i = 0, ..., C$ (assuming matrix indices start at 0), and Q the infinitesimal generator matrix for the process X. The function $I(\cdot)$ is called the large deviations *rate function*, and is continuous and convex.

Figure 3 displays examples of the objective function in (4), and of the large deviation rate $I(A)$ when A varies. Note that

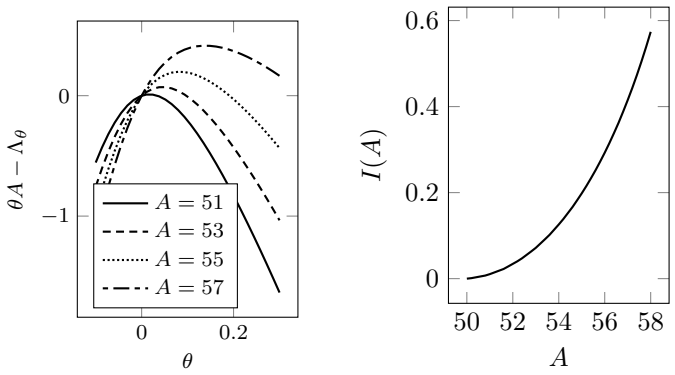


Figure 3. Some values of $\theta A - \Lambda_\theta$ *(left)*, and corresponding large deviation rates *(right)* for the average occupancy during T, when $\lambda = 50$, $\mu = 1$, $C = 65$ (blocking rate for non-deferrable demand: 0.0064).

the large deviation analysis only provides the *rate* $I(A) > 0$ at which the probability tends to 0 as T increases. In this paper we nevertheless intend to ignore the $o(T)$ in (3), or more precisely to ignore its variations with T, and directly use $Ke^{-TI(A)}$ (for an appropriate constant K) as an approximation for the probability of the average occupancy over a period T to exceed A. This will allow us to look for combinations of T and A such that $P_{A,T}$ is small enough. We choose the value of the constant K such that the formula gives a correct response when T tends to 0, hence we will consider that

$$P_{A,T} \approx P_{A,0}e^{-TI(A)}, \tag{5}$$

with $P_{A,0}$ approximating the probability that the instantaneous bandwidth used by non-deferrable traffic exceeds A. For our session model this probability is simply

$$\frac{1}{\sum_{k=0}^{C}\rho^k/k!}\sum_{i=\lceil A\rceil}^{C}\frac{\rho^i}{i!}, \tag{6}$$

which is not continuous in A. For later convenience we will preferably use for $P_{A,0}$ an approximation that is *continuous* in A, for example by taking $P_{A,0}$ as in (6) for integer values of A and piecewise linear between (our choice for the curves plotted in this paper), keeping the difference very small.

The approximation (5) gives us a relationship between T and A: the minimum T such that we can offer some capacity $C - A$ to deferrable service with "failure" probability below ϵ would be

$$T \approx \frac{\log P_{A,0} - \log\epsilon}{I(A)}.$$

Inverting that function in A, the difference $C-A$ is the amount of capacity that can be offered for deferrable service with probability $1-\epsilon$ within delay T during the peak hour, which we denote by $D(T,\epsilon)$:

$$D(T,\epsilon) \approx C - \inf\left\{A : \frac{\log(P_{A,0}/\epsilon)}{I(A)} < T\right\}. \tag{7}$$

As expected, that capacity increases with the guaranteed delay: the rate $I(A)$ increases with A (see Figure 3) while $P_{A,0}$ decreases from (6), hence the inf in (7) decreases with T. Moreover, since $I(A)$ and $P_{A,0}$ vary continuously with A, the right-hand side of (7) is continuous in T, as the inf describes the inverse of the continuous and strictly decreasing function $A \mapsto \frac{\log(P_{A,0}/\epsilon)}{I(A)}$.

An example is displayed in Figure 4, together with simulation results to illustrate that the large deviations theory very accurately predicts the throughput that can be offered to deferrable service as a function of the delay T. More evi-

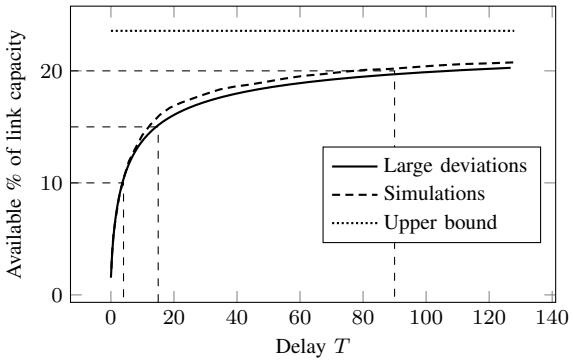


Figure 4. Available capacity for deferrable demand (proportion of the link capacity), with out-of-time probability less than 0.01 (C=65, λ =50, μ=1, blocking probability=0.00645, non-deferrable usage=76%)

dence of this accuracy is given in Figure 5, suggesting that the large deviations approach slightly underestimates the available capacity, with a relative error below 5% (for reasonable link loads) that decreases when the link capacity increases.

Figure 4 shows the case of a link for which the offered non-deferrable traffic (in number of sessions) is 50, but that is dimensioned to $C = 65$ to keep a blocking rate below 0.8%. This results in only 76% of the link capacity being used on average, hence some margin (up to 24% of the link capacity) to offer deferrable service. Both simulation and large-deviation results indicate that we could use 15% of the link capacity–thus reaching 91% link utilization–by proposing a service with delay below $15/\mu$ and a 99% guarantee.

More stringent delay constraints could be preferred: with the same 99% guarantee but for the delay $4/\mu$ we can use 10% of the total link capacity, thus reaching a 86% usage for that link. Alternatively, for very large delays (around $90/\mu$) the link usage rate can get as high as 96%.

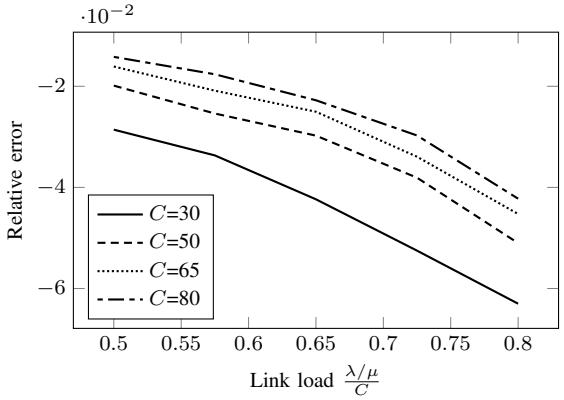


Figure 5. Relative error of (7) to predict deferrable supply with respect to simulations for delay $T = 10$, with out-of-time probability less than 0.01, μ=1.

C. Offering different delay guarantees

Before extending our results to the network case, let us briefly evoke the possibility of proposing simultaneously different "deferrable traffic" offers, with distinct delays and likely with different prices. This provides the network manager with even more flexibility, to segment demand and reach a higher social welfare (and/or higher revenues).

A way to provide that service in practice is to use priorities, traffic with tighter delay constraints having higher priority (and of course, non-deferrable traffic having the highest priority). For the example of Figure 4, as much as 10% of the link capacity can be sold for a "$4/\mu$-delay" service. If that amount is sold, then the network manager can still devote an additional 5% of the link capacity to a "$15/\mu$-delay" service, and even another additional 5% to a "$90/\mu$-delay" service.

This option, and in particular the revenue-maximization possibilities it offers, are not developed in this paper: in the following sections we still consider a unique delay T. But the same simple reasoning as done here is applicable to our next results as well.

IV. A simple network case

In this section we consider the simplest generalization of our results, to a 2-link network topology. We explain how the large-deviation results obtained for one link can be applied for multiple-link transfers, insisting on the importance of caching data in intermediate nodes.

A. Model

Let us consider the simple network topology depicted in Figure 6, with three nodes, two links, and three types of flow: we denote by $X_i(t)$ the number of ongoing non-deferrable sessions using link i only for $i = 1, 2$ at time t, and by $X(t)$ the number of ongoing non-deferrable sessions using both links. Mirroring the previous section, we denote by λ, λ_1 and λ_2 the arrival rates for sessions using both links, link 1 only, and link 2 only, respectively. We assume as before that all sessions use the same bandwidth b, so that the capacity C_i of link i can be expressed as the maximum number of sessions that can simultaneously use that link. Recall that we assume sessions for the three types of flows have the same duration distribution (namely, exponential with parameter μ).

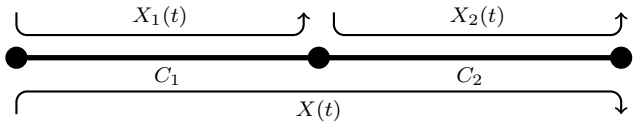


Figure 6. A simple network topology with three types of non-deferrable sessions (represented by arrows). Arc are labelled with their capacity.

B. Available capacity on one link

Under our assumptions, the number of sessions using a given link i is not exactly an M/M/C_i/C_i queue since some requests of two-link connections can be blocked because of the other link. Hence treating $X(t) + X_i(t)$ as an M/M/C_i/C_i queue will be over-pessimistic, but will yield a lower bound of what can be offered as deferrable traffic on that link. Additionally, we can expect this lower bound to be close to the actual value when the blocking probability of non-deferrable flows is low, which is the case in properly dimensioned systems.

Therefore we will take, as an estimate of the available capacity on each link, the result obtained from the analysis in Section III, taking for the arrival rate the sum of the arrival rates of all paths using that link (hence, for our 2-link example, taking $\bar{\lambda}_i = \lambda + \lambda_i$ as the arrival rate on link i, $i = 1, 2$). Figure 7 provides an illustration, where for each link we observe results similar to the one-link case: the large deviation approach provides a very accurate estimation of the available capacity on each link. The gap is a bit larger than in Figure 4, though, especially for link 1, because of the blocking of some two-link sessions due to saturation of link 2 (link 2 has a larger blocking rate than link 1), an aspect neglected in our large deviation approach as explained above.

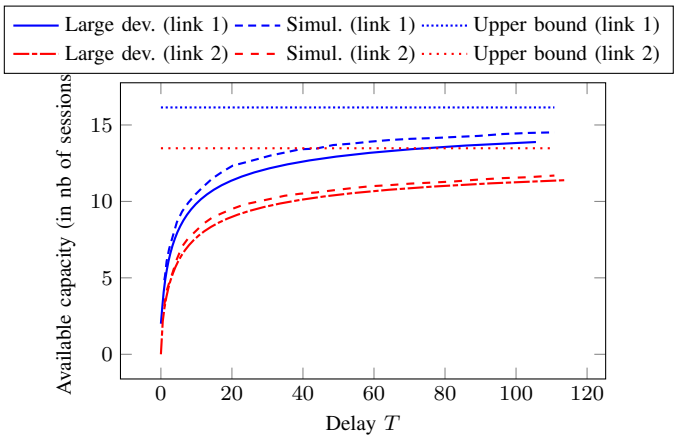


Figure 7. Available capacity for deferrable demand, with out-of-time probability less than 0.01 ($C_1 = 58, C_2 = 60$, $\lambda = 27$, $\lambda_1 = 15$, $\lambda_2 = 20$, μ=1, blocking probabilities$\approx$(0.0035,0.01), non-deferrable usage$\approx$ $(72\%, 78\%)$).

C. Available capacity on a path

Now consider the "long" path in Figure 6. To quantify the amount of bandwidth that could be offered on that path for

deferrable service, we distinguish two cases, according to the possibility or not of caching (storing) data in the middle node.

1) Without caching at the middle node: When no data caching is possible at the middle node, deferrable traffic on the path should be controlled by the source, to send data only when there is capacity available on the whole path, i.e., at an instantaneous rate equal to the minimum of the available rates on the traversed links as proposed in [15]. We leverage here the fact that an SDN architecture can be aware of the usage of each link, and use that knowledge to control the sending rate of each deferrable traffic source. Hence for our two-link path, we are looking for the maximum capacity D such that, assuming the system in stationary regime at time 0,

$$\mathbb{P}\left(\frac{1}{T}\int_{t=0}^{T}\min(C_1-X_1-X\ ,\ C_2-X_2-X)\mathrm{d}t<D\right)\leq\epsilon, \quad (8)$$

where we omit the dependence on t of X_1, X_2, and X: at time t, $X_i(t)$ is the number of non-deferrable sessions using link i only, and $X(t)$ the number of non-deferrable sessions on the two-link path. The left-hand side of (8) being continuous in D, we actually have equality in (8) for the optimal D, that we denote by D_{path}.

We can now state a result lower-bounding D_{path} to the value obtained when no non-deferrable traffic uses the two-link path.

Proposition 1: The available capacity D_{path} on the two-link path is lower-bounded by the one obtained when only one-link sessions arrive, with arrival rate $\bar{\lambda}_i = \lambda_i + \lambda$ on link $i = 1, 2$.

The proof is provided in Appendix A.

Proposition 1 is illustrated by simulations in Figure 8, where we plot the available transmission rates on the two-link path when arrival rates of non-deferrable sessions on link-1, link-2, and the two-link paths are respectively $\lambda_1 + \beta\lambda$, $\lambda_2 + \beta\lambda$, and $(1-\beta)\lambda$, for β varying in $[0, 1]$. As stated in the proposition,

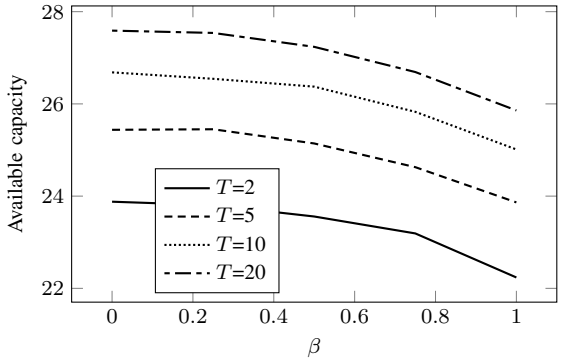


Figure 8. Available capacity for deferrable demand, with out-of-time probability less than 0.01 when $\lambda_1 = 10 + 10\beta$, $\lambda_2 = 15 + 10\beta$, $\lambda = 10(1-\beta)$, $\mu = 1$, C_1=40, C_2=45 (simulation results).

the available rates are the lowest when $\beta = 1$.

Unfortunately, we do not have a large-deviation derivation for that case for a general delay T. However we think that a networked version of the deferrable service *should* involve caching in intermediate nodes to reach a significant use of network links. As an illustration, consider a chain of M links behaving as independent and identical M/M/C/C queues with offered traffic ρ on each link. Then, without caching, the steady-state probability that at least some capacity D is available on the whole path equals $U(D)^M$, with

$$U(D) = \sum_{i=0}^{C-\lceil D\rceil} \frac{\rho^i/i!}{\sum_{k=0}^{C}\rho^k/k!} < 1,$$

and therefore decreases exponentially in M. When $T \to \infty$, the maximum capacity that could be offered on the M-link path equals the average minimum available capacity among the M links, that can be computed as $\sum_{D=1}^{C} U(D)^M$. For $T < \infty$ we can of course offer even less.

Figure 9 plots this upper bound for some example values, showing that the available bandwidth for the deferrable service decreases very fast with M, hence a very limited service offer even for numbers of hops around 5, a reasonable value [9]. Therefore, we think a network application of the deferrable

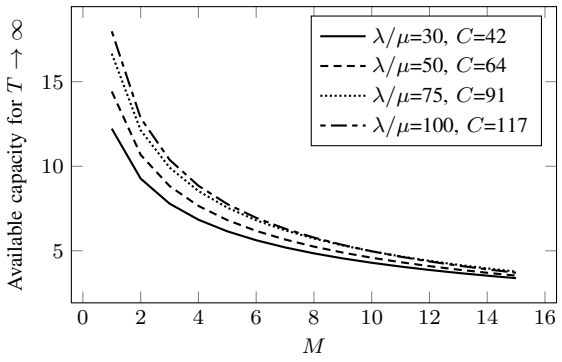


Figure 9. Available capacity on an M-link path (multiple of the session rate b) without caching, when all links behave as independent M/M/C/C queues. Link capacity C is optimized to maintain blocking rate below 0.01.

service is worth considering only when caching is available at intermediate nodes. This is also illustrated later, in Figure 10.

2) With caching at the middle node: With the possibility of caching data in the middle node, the deferrable service does not need to limit instantaneous data rates to the minimum of the instantaneous available rates on the path links: data can be sent on a per-link basis, just being constrained by the currently used link instantaneous available capacity, and is then possibly cached at the next hop. The capacity of interest then becomes the minimum (over links) *average* (over time) available capacity on a period of length T. Mathematically, while without caching we were looking for D_{path} such that

$$\mathbb{P}\left(\frac{1}{T}\int_{t=0}^{T}\min(C_1-X_1-X\ ,\ C_2-X_2-X)\mathrm{d}t<D_{\text{path}}\right)=\epsilon,$$

with caching we are looking for $D^{\text{c}}_{\text{path}}$ such that

$$\mathbb{P}\left(\frac{1}{T}\min\left\{\int_{t=0}^{T}C_1-X_1-X\mathrm{d}t,\ \int_{t=0}^{T}C_2-X_2-X\mathrm{d}t\right\}<D^{\text{c}}_{\text{path}}\right)=\epsilon, \quad (9)$$

which will give larger available capacities, i.e., $D^{\text{c}}_{\text{path}} \geq D_{\text{path}}$, since the minimum of averages is larger than the average of minimums.

More specifically, we claim that with caching, the available capacity on the two-link path is very close to the minimum of

the available capacities along the path, computed separately with the common delay target T.

The reasoning is as follows: the principle of large deviations not only gives the rate at which the probability of "exceptionally large average occupancy" on each link decreases with the considered duration T, but also indicates *how* such large occupancies can be attained. Specifically, only the most likely behaviors leading to such large average occupancies should be considered.

In an M/M/C/C queue, one can show that due to the convexity of the rate function, the most likely trajectories yielding to a given high average occupancy are those with a (almost) constant occupancy, equal to that average. The intuition is that trajectories going below that level must also have periods with even higher occupancy (to reach the same average value), which have a high "likelihood cost" since the likelihood of having an extra client (i.e., an arrival rather than a departure) decreases with the occupancy.

Going back to our two-link path, the most likely way to have "exceptionally bad" performance on the path is to have only one link with "exceptionally large average occupancy", more specifically, the one for which such occupancy is the most likely. But when the target probability of those exceptional events is ϵ, this is precisely the link i with the smallest available capacity D_i computed from (7) for link i. Then, the most likely behavior for the other link is to have more than D_i available.

This reasoning leads to the simple method below.

Method 1: To estimate the available capacity on the two-link path with caching, take the minimum available capacity of both links, computed independently from (7), with a session arrival rate $\bar{\lambda}_i = \lambda + \lambda_i$ on link $i = 1, 2$.

Figure 10 displays an example for a symmetric ($C_1 = C_2$ and $\lambda_1 = \lambda_2$) and "pessimistic" case ($\lambda = 0$), showing that our large-deviation results applied separately to each link still capture very accurately the variations of what can be offered end-to-end with the tolerable delay. Figure 10 also shows the importance of caching: without caching, Method 1 does not apply and the available capacity on the path is significantly below the available capacity on each link. Note that for that case the upper bound as T increases is consistent with the observations in Figure 9 for $M = 2$.

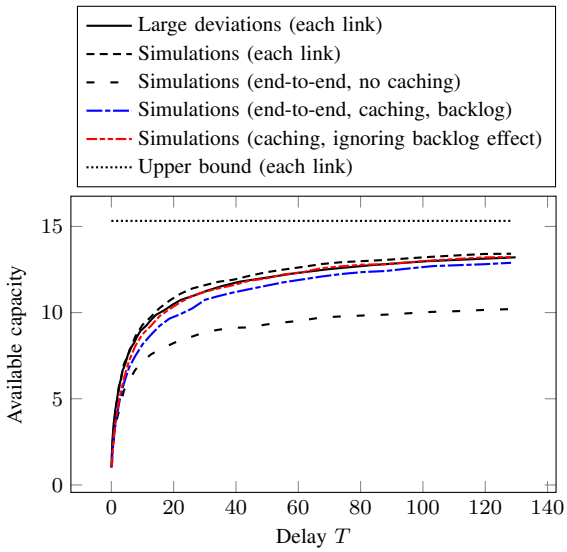


Figure 10. Available capacity for deferrable demand, with out-of-time probability less than 0.01. (Parameters: C_1=C_2=65, λ=0, λ_1=λ_2=50, μ=1, blocking probabilities$\approx$ 0.0064, non-deferrable usage$\approx$ 76%)

A direct consequence of Method 1 is that the use of the available "delay-T" capacity on the links can be on any path, leading to a straightforward method to check feasibility of a deferrable-traffic matrix:

Method 2: To check whether a deferrable traffic throughput profile R_1, R_2, R (on the link-1, the link-2, and the two-link paths respectively) can be served with the delay guarantee T and the out-of-time probability ϵ, verify that the link capacity constraints $R_1 + R \leq D_1$ and $R_2 + R \leq D_2$ are satisfied, with $D_i, i = 1, 2$, obtained as in Method 1.

3) Possible loss of efficiency: The expression (9) actually forgets a part of the problem, by just focusing on the average bandwidth available along the path: there may indeed be cases when some bandwidth is available on link 2 *before* the equivalent amount is available on link 1. In that case, even if link 2 is the bottleneck in the sense of Method 1, not all the capacity of link 2 can be used, hence some possible loss with respect to the proposition due to this "backlog effect", as simulation results show in Figure 10.

However, we think this effect should be minor in practice because of the pipelining that occurs: recall that we have been pessimistic in Section III by considering that no deferrable data received for treatment in the interval $[t-T, t]$ was treated in that interval. This is how we reached (1), and which is simulated in Figure 10. Additionally, Figure 10 considers a worst-case situation, where both links have the same available capacity: this maximizes the likelihood of data being backlogged by link 1 among situations where link 2 is the bottleneck. Even with those two pessimistic assumptions the effect is not so salient, we expect it to be even less important in practice and therefore ignore it in the remainder of this paper.

V. Building a "delay-T network"

In this section, we propose to extend the results of the two previous sections over a whole network. More specifically, we suggest that the manager of an existing network decide on a delay T for the deferrable service, and we provide a methodology to estimate the capacities that could be offered with that delay constraint. We first extend Method 1 to claim that an analysis on a per-link basis is sufficient: the delay guarantee on each link will still be satisfied end-to-end. Hence we can just represent a "delay-T network" as a network with the same topology as the original one, with some "delay-T capacity" on each link.

Note that while our "delay-T network" comes at no costs in terms of transmission capacities, there may be some storage costs at the network nodes to provide caching as described in the previous section. We expect the storage amounts to remain small because of pipelining of data treatment, but quantifying the amount of storage space needed is of interest and should be studied in future work. Here, we assume that storage is cheap and focus on transmission capacities.

A. Lower-bounds: assuming independence among links

In the rest of this section we assume sufficient caching is available within the network. As in Method 1, we target delay guarantees on an end-to-end basis. The reasoning is exactly the same: given a path, considering all traversed links as independent (with arrival rates equal to the sum of the arrival rates of all flows using that link), should leave less capacity than the initial setting. We will use the lower bound obtained this way as an estimate of the available capacities on links.

Then, as in the previous section, we exploit the properties of large deviations as depending only on the most likely trajectories, to claim it is sufficient to consider the minimum available capacity (for the chosen delay T and guarantee level ϵ) among the path links. Indeed, again the most likely way to get bad average performance over a sufficiently long period T is through the "weakest" link in the path, i.e., the most saturated. And for that path, the most likely trajectory leaving a capacity D on average is one leaving a (almost) constant capacity D; for the other links the most likely behavior would not be far away from the average, hence leaving at least D except for very short durations (managed through caching, and only slightly affecting the delay for deferrable service).

Method 3: Assume that there are sufficiently large caching capacities in intermediate nodes in the network, and consider a single path on that network. Then, to estimate the available capacity on any path, take the minimum available capacity of the links on that path, computed independently from (7), with an arrival rate equal to the sum of all arrival rates for sessions using that link.

B. How much capacity to offer?

Treating all links as independent has the advantage of removing complex delay constraints due to multi-link paths: to get some delay-T capacity D over a path, one just needs to ensure to get delay-T capacity on each link over that path, i.e., exactly as in the initial network for non-deferrable traffic. We therefore have the counterpart of Method 2:

Method 4: To check whether a deferrable-traffic throughput profile (on all possible paths on the network) can be served with the delay guarantee T and the out-of-time probability ϵ, verify that the link delay-T capacity constraints on all links are satisfied, where those capacities are obtained independently on each link from (7), taking for the arrival rate the sum of all arrival rates for non-deferrable flows using that link.

A simple way of representing the delay-T service is therefore to keep the network topology, and display the available capacity on each link for the chosen delay T. An example is provided in Figure 11, for two different values of the delay.

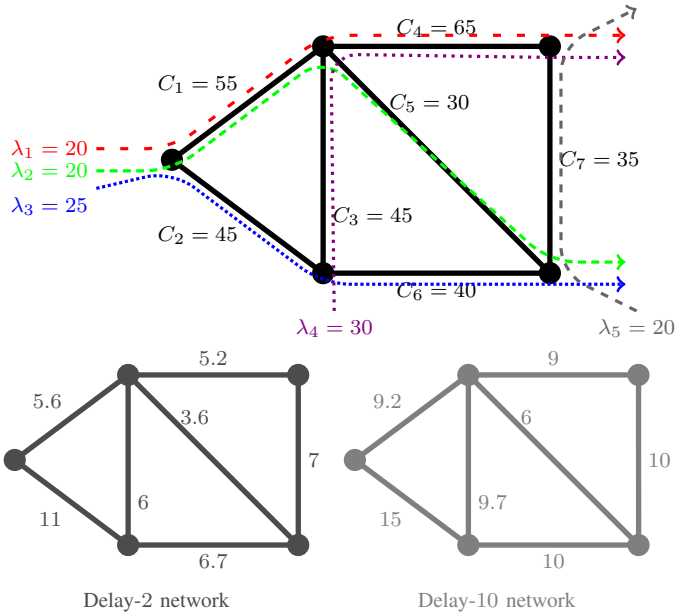


Figure 11. An example of network topology with existing non-deferrable demands *(top)*, and the associated delay-T network for $T = 2$ and $T = 10$, when $\epsilon = 0.01$ and $\mu = 1$. Arcs are labeled with their capacities; all link blocking rates are below 1%.

VI. Conclusion and Perspectives

Telecommunication networks are over-dimensioned with respect to the average traffic they carry, because of traffic demand variations. In this paper, we propose to leverage these extra capacities to provide a new service, using only the resources left available by the non-deferrable traffic. We show that we can still provide guarantees for the delay experienced by such traffic, and provide a methodology based on large deviations analysis to estimate the capacities of the corresponding deferrable-service network, a "new" network that does not imply any capacity expansion costs but possibly some in-network storage costs. Even if a new external service is not offered, the ideas discussed in this paper can be used for internal purposes by a large operator. For example, large operators periodically have to perform some synchronization or backup of large distributed databases, which is very bandwidth-consuming. Although we suspect they already perform those operations using low-priority traffic, our results help understand the type of delays that could be guaranteed, or reciprocally the maximum loads of such low-priority traffic that could be supported while keeping delays reasonable.

Possible future work includes a quantitative study of the amount of in-network storage needed to make the most of such a system: we have assumed that there is sufficient caching space in the network, and would be able to estimate the associated costs to gain even more insight regarding the realizability of our proposition. Another interesting extension is to consider heterogeneous access rates among users: our model (with equal access capacities for all users) provides useful insights, but given the diversity of available access technologies it would be more realistic to study different types of sessions, with different capacities and probably different duration distributions. A first step could be to assume an access rate that depends only on the entry node: in Figure 1 we would have a common b_j for all users accessing the network through entry point j. The case of routing (for non-deferrable traffic) that would depend on the current network conditions is also worth considering: we have ignored it here for simplicity, so that we have Poisson arrivals for each type of route, but in practice we may have spill-over sessions on secondary routes.

APPENDIX A
PROOF OF PROPOSITION 1

Proof: Let us consider the process $(X_1 + X, X_2 + X)$, and consider a slightly different Markov process $(\tilde{X}_1, \tilde{X}_2)$ such that arrivals of one-link sessions are unchanged but arrivals of 2-link sessions are "duplicated" into a one-link session on each link. Mathematically, for arrivals we have transitions
- $(\tilde{X}_1, \tilde{X}_2) \to (\min(C_1, \tilde{X}_1 + 1), X_2)$ with rate λ_1,
- $(\tilde{X}_1, \tilde{X}_2) \to (X_1, \min(C_2, \tilde{X}_2 + 1))$ with rate λ_2,
- $(\tilde{X}_1, \tilde{X}_2) \to (\min(C_1, \tilde{X}_1{+}1), \min(C_2, \tilde{X}_2{+}1))$ with rate λ.

In terms of departures, all sessions of $(\tilde{X}_1, \tilde{X}_2)$ leave after independent exponentially distributed times with parameter μ (i.e., the "duplicated" sessions are then independent).

Then $\min(C_1 - \tilde{X}_1, C_2 - \tilde{X}_2)$ is stochastically smaller than $\min(C_1 - (X_1{+}X), C_2 - (X_2{+}X))$, since the differences are:
i) in the original case more sessions are blocked: when a link is saturated and a 2-link session arrives, the state is unchanged while in the new case there is a new session on one link.
ii) in the original case, two-link sessions leave after an exponentially distributed time with parameter μ, freeing one "server" (the space for one session) simultaneously on both links. In contrast, in the new case the duplicated sessions leave one by one, each one with an exponentially distributed time with parameter μ.
Hence there tends to be more active sessions in the new case than in the original one, thus less space for deferrable flows.

Finally, consider another process $(\bar{X}_1, \bar{X}_2)$, that only differs from $(\tilde{X}_1, \tilde{X}_2)$ in that the "duplicated" sessions now arrive independently (hence we have independent arrivals on each link according to two independent Poisson processes with rate λ for those specific sessions). In summary, $\bar{X}_1$ and $\bar{X}_2$ are simply two *independent* processes, each $\bar{X}_i$ $(i = 1, 2)$ corresponding to an M/M/C_i/C_i queue with arrival rate $\bar{\lambda}_i$ and service rate μ.
Now remark that in both cases, for any fixed $i \in \{1, 2\}$ the "marginal" processes $\bar{X}_i$ and $\tilde{X}_i$ both correspond to an M/M/C_i/C_i with the same arrival rate $\bar{\lambda}_i$ and service rate μ, hence are stochastically equivalent. But because of some joint arrivals (the duplicated ones) in the case of $(\tilde{X}_1, \tilde{X}_2)$, the processes $\tilde{X}_1$ and $\tilde{X}_2$ are positively correlated.
It results that $(C_1 - \bar{X}_1, C_2 - \bar{X}_2)$ and $(C_1 - \tilde{X}_1, C_2 - \tilde{X}_2)$ also have marginal processes that are stochastically equivalent, but $C_1 - \tilde{X}_1$ and $C_2 - \tilde{X}_2$ are positively correlated while $C_1 - \bar{X}_1$ and $C_2 - \bar{X}_2$ are independent.

Let us now define, for $i = 1, 2$, $p_i(\delta) := \mathbb{P}(C_i - \bar{X}_i < \delta)$. Then for any $\delta > 0$:
- because of the independence between $\bar{X}_1$ and $\bar{X}_2$ we have $\mathbb{P}\left(\min(C_1 - \bar{X}_1, C_2 - \bar{X}_2) < \delta\right) = p_1(\delta) + p_2(\delta) - p_1(\delta)p_2(\delta)$;
- now since $\tilde{X}_i$ is stochastically equivalent to $\bar{X}_i$ for $i = 1, 2$, we have at each instant

$$\mathbb{P}\left(\min(C_1 - \tilde{X}_1, C_2 - \bar{X}_2) < \delta\right) = p_1(\delta) + p_2(\delta) - \mathbb{P}\left(\{C_1 - \tilde{X}_1 < \delta\} \cap \{C_2 - \tilde{X}_2 < \delta\}\right).$$

But because of the positive correlation between $C_1 - \tilde{X}_1$ and $C_2 - \tilde{X}_2$, the probability that *both* $C_1 - \tilde{X}_1$ and $C_2 - \tilde{X}_2$ exceed δ is larger than if those processes were independent:

$$\mathbb{P}\left(\{C_1 - \tilde{X}_1 < \delta\} \cap \{C_2 - \tilde{X}_2 < \delta\}\right) \geq p_1(\delta)p_2(\delta).$$

Hence $\min(C_1 - \bar{X}_1, C_2 - \bar{X}_2)$ is stochastically smaller than $\min(C_1 - \tilde{X}_1, C_2 - \tilde{X}_2)$, which yields

$$\begin{aligned}
&\mathbb{P}\left(\frac{1}{T}\int_{t=0}^{T} \min(C_1 - \bar{X}_1(t)\ ,\ C_2 - \bar{X}_2(t))\mathrm{d}t < D_{\text{path}}\right) \\
&\geq \mathbb{P}\left(\frac{1}{T}\int_{t=0}^{T} \min(C_1 - \tilde{X}_1(t)\ ,\ C_2 - \tilde{X}_2(t))\mathrm{d}t < D_{\text{path}}\right) \\
&\geq \mathbb{P}\left(\frac{1}{T}\int_{t=0}^{T} \min(C_1 - X_1 - X, C_2 - X_2 - X)\mathrm{d}t < D_{\text{path}}\right) = \epsilon,
\end{aligned}$$

thus we cannot offer more than D_{path} to the system with arrival rates $\lambda_1 = \bar{\lambda}_1, \lambda_2 = \bar{\lambda}_2, \lambda = 0$. Hence the proposition. ∎

REFERENCES

[1] S. Agarwal, M. Kodialam, and T. V. Lakshman. Traffic engineering in software defined networks. In *Proc. of IEEE INFOCOM*, 2013.

[2] K. Angrishi. An end-to-end stochastic network calculus with effective bandwidth and effective capacity. *Computer Networks*, 57(1):78–84, 2013.

[3] D. Bertsimas, I. C. Paschalidis, and J. N. Tsitsiklis. On the large deviations behavior of acyclic networks of G/G/1 queues. *The Annals of Applied Probability*, 8(4):1027–1069, 1998.

[4] E. Bitar and S. Low. Deadline differentiated pricing of deferrable electric power service. In *Proc. of IEEE CDC*, 2012.

[5] E. Bitar and Y. Xu. Deadline differentiated pricing of delay-tolerant demand. http://arxiv.org/abs/1407.1601, 2015.

[6] C.-S. Chang. *Performance Guarantees in Communication Networks*. Springer, 2000.

[7] F. Ciucu, A. Burchard, and J. Liebeherr. Scaling properties of statistical end-to-end bounds in the network calculus. *IEEE/ACM Trans. Networking*, 14(6):2300–2312, 2006.

[8] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. of ACM SIGCOMM*, 2003.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proc. of ACM SIGCOMM*, 1999.

[10] Bolch. G., S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 2006.

[11] H. Kim and N. Feamster. Improving network management with software defined networking. *IEEE Comm. Mag.*, 51(2):114–119, 2013.

[12] A. Krifa, C. Barakat, and T. Spyropoulos. Optimal buffer management policies for delay tolerant networks. In *Proc. of IEEE SECON*, 2008.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *ACM SIGCOMM Comp. Comm. Rev.*, 38(2):69–74, 2008.

[14] S. Meyn, P. Barooah, A. Bušić, and J. Ehren. Ancillary service to the grid from deferrable loads: the case for intelligent pool pumps in Florida. In *Proc. of IEEE CDC*, 2013.

[15] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal. Fastpass: A centralized "zero-queue" datacenter network. In *Proc. of ACM SIGCOMM*, 2014.

[16] A. Shwartz and A. Weiss. *Large Deviations for Performance Analysis*. Chapman & Hall, 1995.

[17] A. L. Stolyar and K. Ramanan. Largest weighted delay first scheduling: Large deviations and optimality. *The Annals of Applied Probability*, 11(1):1–48, 2001.

[18] S. R. S. Varadhan. Large deviations. *The Annals of Probability*, 36(2):397–419, 2008.

[19] A. V. Vasilakos, Y. Zhang, and T. Spyropoulos. *Delay Tolerant Networks: Protocols and Applications*. CRC Press, 2011.

[20] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.

Demand-Aware Centralized Traffic Scheduling in Wireless LANs

Sangyup Han
KAIST

Myungjin Lee
University of Edinburgh

Myungchul Kim
KAIST

***Abstract*—A heavy deployment of IEEE 802.11 Wireless LANs and limited number of orthogonal channels make lots of Access Points (APs) overlap their interference regions, which greatly increases interferences between APs and stations. In order to cope with the performance degradation caused by the interferences, we propose CO-FI, a centralized Wi-Fi architecture that effectively coordinates downlink transmissions by APs and improves network performance in terms of throughput and end-to-end delay. CO-FI adaptively allocates time slots for APs and stations based on both traffic demands on the stations and a conflict graph that represents interference relationships among the devices. The scheme allows APs in exposed node relationship to use the channel simultaneously by setting the same backoff time. It also effectively avoids downlink conflicts created by hidden node and non-hidden/non-exposed node, by allocating non-overlapping time slots to interfering stations. To implement these adaptive traffic schedules, we design CoMAC, a hybrid MAC protocol at APs. Our evaluation results show that when APs are densely deployed and the network is highly loaded, the scheme achieves 3-5 times more throughput gain than Centaur, a state-of-the-art scheme while its end-to-end delays are 10-90% lower than those of Centaur and CSMA/CA.**

I. INTRODUCTION

IEEE 802.11 Wireless LAN (WLAN) is one of the most popular wireless communication technologies developed so far. Its tremendous success has led to the dense deployment of WLANs almost everywhere. However, the high density also incurs interferences more frequently among wireless Access Points (APs) and devices (or stations) [1]. Hence, more APs may do more harm than good, and hamper the optimal performance of WLANs [2].

In fact, the interference problem in WLANs is one of well-studied topics in the literature. A large body of research work [3], [4], [5] has focused on reducing interference level to improve stations' throughput. In [5], it is seen as a channel allocation problem, and several graph coloring algorithms are explored. In [3], dynamic transmission range control is attempted. However, the heavy deployment of WLANs still creates various interference situations and makes those approaches less effective. Consequently, recent approaches [6], [7], [8] explore centralized traffic scheduling in order to fundamentally minimize the degree of interference.

However, centralized scheduling in general entails high scheduling complexity. Existing solutions therefore trade scheduling granularity for reduced complexity. For instance, Centaur [6], one of the state-of-the-art approaches, performs centralized scheduling only for traffic of hidden and exposed nodes whereas it delegates the scheduling of traffic for non-hidden/non-exposed nodes to Distributed Coordination Function (DCF) of CSMA/CA. Thus, contentions can hurt throughput for traffic destined to the non-hidden/non-exposed nodes. Worse, in the presence of automatic rate adaptation [9], [10], contentions may force selection of lower rates more often, which may further exacerbate performance.

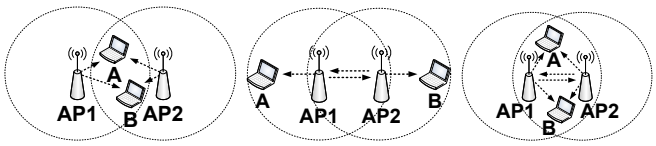
Such trade-off of the state-of-the-art solution eliminates the possibility of improved throughput through precise scheduling. As such, we take into account all of the interference types including non-hidden/non-exposed node for traffic scheduling. To amortize the increased complexity, we only focus on batch-scheduling of high-volume traffic as scheduling low-volume traffic well does little for overall performance improvement.

In this paper, we present *Coordinated* Wi-Fi (CO-FI) that achieves high throughput and low scheduling complexity in WLANs administered by a single authority. CO-FI is designed in a way that a centralized controller computes frame transmission schedules for each AP, and APs run a hybrid MAC protocol called *CoMAC* that can select DCF and Time Division Multiple Access (TDMA) modes flexibly. CoMAC runs in TDMA mode to transmit traffic in a batch fashion scheduled by the controller whereas it runs in DCF mode for transmitting low-volume traffic. Our scheme only schedules downlink traffic as the volume of downlink traffic takes a dominant portion in WLANs [11], [12], [13]. That is, stations access wireless medium in a typical CSMA/CA manner for uplink transmission.

In summary, this paper makes the following contributions:

- We present CO-FI, a novel centralized traffic scheduling mechanism for WLANs that effectively coordinates downlink transmission of frames to stations. The scheme employs demand-aware traffic scheduling. In the scheme, traffic to bandwidth-hungry stations is precisely scheduled while transmission of low-volume traffic takes place opportunistically. This allows CO-FI to keep scheduling overhead low as a small number of stations need to be scheduled on average.
- We design CoMAC, a *hybrid MAC scheme* that can elastically switch between DCF and TDMA. CoMAC works in TDMA mode when the transmission of high-volume traffic strictly follows the schedule of the centralized scheduler. In contrast, DCF mode is activated for opportunistically scheduling the transmission of low-

ISBN 978-3-901882-83-8



(a) (b) (c)

Fig. 1: Types of interferences: (a) Hidden-Node (HN), (b) Exposed-Node (EN) and (c) Non-Hidden/Non-Exposed Node (NHNEN). *Stations A* and *B* are associated with *AP1* and *AP2*, respectively.

volume traffic, which prevents starvation. Because the scheme supports both modes, potential schedule conflicts in TDMA mode can be addressed using DCF. This not only allows our scheme to avoid complex rescheduling but also makes it robust to errors in time synchronization and traffic demand estimation.

- Our extensive simulation results demonstrate that CO-FI outperforms Centaur, the most well-known solution, and CSMA/CA when APs are densely deployed and the network is highly loaded. Specifically, CO-FI achieves 3-5× higher throughput than both schemes and its end-to-end delays are 10-90% lower than those of Centaur and CSMA/CA.

The remainder of this paper is organized as follows. Section II introduces the basic concepts necessary when discussing the design of CO-FI in Section III. In Section IV, we present evaluation results. Section V discusses related work before we conclude in Section VI.

II. PRELIMINARIES

In this section, we discuss three primary concepts—*interference types*, *time window* and *time slot*—that are the basis in devising our scheme.

Interference types. We first explore different characteristics of interferences between wireless links. For this, we adopt a well-known data structure called *Conflict Graph* [14]. A vertex in a conflict graph represents an AP or a station, and a directed edge between two vertices means a wireless link. If there is a wireless link (same as an edge in a conflict graph), it means that a signal from an AP (or station) can successfully be transmitted to the other AP (or station).

Figure 1 illustrates three types of interferences. *Stations A* and *B* in the figure are associated with *AP1* and *AP2*, respectively. In addition, the wireless links outgoing from the APs are only shown in Figure 1 since we only consider the interferences caused by downlink transmissions. A wireless link from node i to node j is denoted as L_{ij}. For instance, L_{1A} represents the wireless link from *AP1* to *Station A*; other wireless links are denoted in the same manner.

Although the concepts of hidden-node and exposed-node problems are well known, the way to identify them from a conflict graph varies across studies [14], [15]. Thus, we slightly modify them and use the following equations when the edge set, *E*, of a conflict graph is given:

- Hidden-Node (HN) interference:

$$\{L_{12},\, L_{21}\} \not\subset E \textbf{ and } \{L_{1A},\, L_{2B}\} \subset E \textbf{ and } (L_{1B} \in E \textbf{ or } L_{2A} \in E), \quad (1)$$

- Exposed-Node (EN) interference:

$$(L_{12} \in E \textbf{ or } L_{21} \in E) \textbf{ and } \{L_{1A},\, L_{2B}\} \subset E \textbf{ and } (L_{1B} \notin E \textbf{ and } L_{2A} \notin E), \quad (2)$$

- Non-Hidden/Non-Exposed Node (NHNEN) interference:

$$(L_{12} \in E \textbf{ or } L_{21} \in E) \textbf{ and } \{L_{1A},\, L_{2B}\} \subset E \textbf{ and } (L_{1B} \in E \textbf{ or } L_{2A} \in E). \quad (3)$$

Note L_{uv} represents an edge from vertex u to vertex v. We use number for u and v to denote APs and use alphabets to denote stations. In addition, further notice that Eqs. 1, 2, and 3 cover the case that only one of the two APs senses the other AP, which is caused by the asymmetric nature of wireless medium. If edges meet none of the above conditions, we treat them *as if there is no interference among them*. These definitions are applied to more complex WLANs through pairwise comparisons of edges iteratively.

Issues with these interferences: The HN and NHNEN interferences result in collision at APs when simultaneous transmissions take place from APs to stations. This therefore causes retransmissions and even frame drops. While DCF may mitigate the impact of these interferences, not all collisions can be avoided. Moreover, frame collisions may make the APs decrease their PHY transmission rate, which results in performance degradation. On the other hand, if stations experience EN interference, their associated APs can benefit from simultaneous transmissions and achieve improved throughput. However, if an AP senses the signal of other APs, it defers its transmission and fails to exploit the EN interference.

Time window and slot. A time window is a basic unit of scheduling frame transmissions, and a time slot is a constituent of a window. Note that a frame transmission can span multiple consecutive time slots due to a low transmission rate. In our paper, the duration of a window, Δ, is set to 20 ms, and each window consists of 800 slots. Thus, one slot corresponds to 25 μs. Whilst both variables are flexibly configurable, we choose them empirically while running simulations.

III. CO-FI DESIGN

We now design CO-FI, a centrally-coordinated WLAN architecture. In CO-FI, a centralized controller coordinates traffic transmission schedules of APs in order to maximize throughput and to minimize end-to-end delay in the WLANs. We first present an overview of CO-FI, and then describe each component that constitutes the architecture.

A. Overview

CO-FI adopts a centralized coordination model where a controller precisely schedules traffic transmission timings of APs. Under this model, APs communicate with the controller to form a control loop for traffic transmission coordination.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

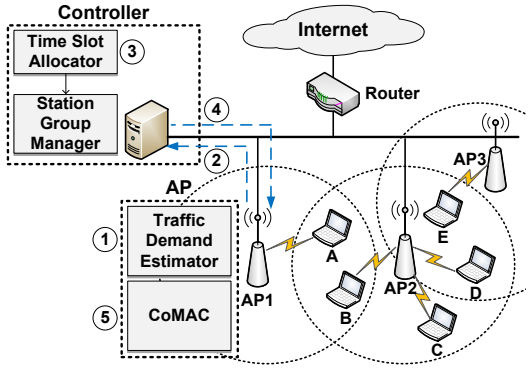


Fig. 2: Overview of CO-FI. The overall procedure is to: (1) estimate future traffic demands to stations; (2) inform the controller of the traffic estimates for stations; (3) compute time slot allocation schedule for the traffic among interfering stations; (4) distribute the schedule to APs from the controller; (5) transmit frames either based on time slots allocated or opportunistically if no time slot is allocated.

Specifically, the controller dictates when each AP can transmit, and the APs abide by the controller's instruction. CO-FI adaptively allocates time slots for APs while being aware of the amount of workloads that arrive at each AP. In addition, CO-FI addresses potential schedule conflicts by leveraging existing CSMA/CA. We call this strategy CoMAC which is discussed in Section III-D.

We describe CO-FI's working mechanism through a simple scenario illustrated in Figure 2. Suppose that the amount of traffic to *Stations A* and *B* is 120 Kbits and 60 Kbits, respectively, over a window Δ. *AP1* and *AP2* estimate future traffic demands by combining the size of existing packets in their buffer with the amount of incoming traffic (**Step 1** in Figure 2). The APs next ship these estimates to the controller (**Step 2**). The controller then allocates time slots in proportion to these estimates as the two APs compete for the same wireless medium (**Step 3**). In Figure 2, because the two links are in an HN relationship, the controller allocates non-overlapped time slots in proportion to each AP's demand (*i.e.*, 533 slots to *A* and 267 slots to *B*). This prevents the APs from simultaneous transmission. In addition, if the estimated traffic volume to a station is lower than a threshold, ψ (further discussed in Section III-D), the station is excluded from scheduling and allowed to do the typical opportunistic medium access via CSMA/CA. This effectively reduces scheduling complexity. The controller distributes the allocation information to APs (**Step 4**). The APs then take one of the following two actions (**Step 5**): i) if a destination station is allocated to some time slots, a frame to the station can only be transmitted within its time slots; ii) for those stations without allocated time slots frames are transmitted opportunistically via CSMA/CA.

One key advantage of our scheme is that it requires no modification of the stations, which renders it practical in deploying it into the existing and future wireless networks. In the rest of this section, we discuss how we design each component of CO-FI in Figure 2.

B. Traffic Demand Estimator at APs

APs estimate how much amount of traffic the APs should transmit for a given station at the next time window. At the end of every time window, the APs conduct the process for each station associated with them. They then send the estimates to the controller for a centralized time slot allocation process.

The APs maintain a table that consists of the following column elements: (W_i, *dstMAC*, T_i) where W_i denotes time window i, *dstMAC* means the MAC address of a destination station, and T_i is a total of traffic demand (in bytes) to the destination in W_i. Whenever the APs see a new frame, APs extract *dstMAC* and frame size f and update T_i with f. (*i.e.*, $T_i \leftarrow T_i + f$). For a given *dstMAC*, we refer to the information stored in the table and estimate the future demand as follows:

$$FD_i^{dstMAC} = \min(MA_i + D_i, Tx_i \times \Delta), \tag{4}$$

where MA_i is an exponential moving average of incoming traffic to the station at W_i, D_i is the traffic amount (in bytes) in the buffer for the station at W_i, and Tx_i is the current PHY transmission bitrate for the station at W_i. Note that in (4), Δ is converted from millisecond to second. These three variables (MA_i, D_i, Tx_i) are maintained on a per-station basis.

In the $\min$ function of the equation, the left-hand side term indicates the amount of traffic that the AP should transmit at the next time window. However, when Tx_i is low, the AP cannot transmit all the traffic within the next time window. Therefore, we put the right-hand side term as an upper bound. MA_i is calculated using the following equation:

$$MA_i = \begin{cases} \alpha \cdot T_i + (1-\alpha) \cdot MA_{i-1}, & \text{if } T_i \neq 0 \\ (1-\alpha) \cdot MA_{i-1}, & \text{otherwise} \end{cases}, \tag{5}$$

where T_i is the amount of traffic received during time window i, and α is the coefficient that represents the degree of weighting the current traffic. We set α to 0.8 in this paper.

While in principle APs can report the computed demands at every window ($\Delta = 20\,ms$), in practice we have APs report $\max(FD_i^{dstMAC}, FD_{i-1}^{dstMAC})$ every other window (so, 40 ms). We do this because we found that reporting the traffic demand at every window sometimes became unstable, and reporting at every other window achieved the highest throughput (always 3% greater than the former in our test scenarios).

C. Controller functions

The CO-FI controller has two core functions: time slot allocation and station grouping. In addition, the controller synchronizes time among APs by using Network Time Protocol (NTP). This protocol is known to make a few millisecond synchronization precision possible in the wired local networks [16]. To tolerate that level of synchronization error, in our design we use a large (i.e., $20ms$) window.

Time Slot Allocator. Figure 3 illustrates the time slot allocation procedure. At first, APs estimate traffic demands for stations and send them to a controller every $2 \cdot \Delta$ (**Step 1** in the figure). The controller receives the estimates, as the form of (station's MAC address, traffic estimate), from the APs for

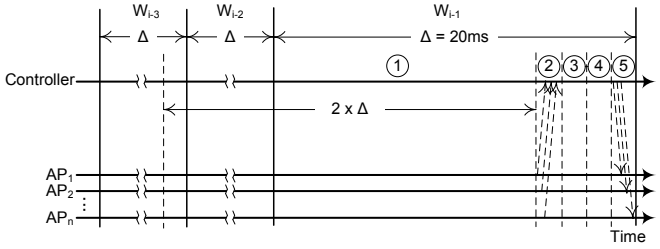


Fig. 3: Slot allocation scheduling procedure: (1) APs estimate traffic demand for stations for $2{\cdot}\Delta$ period, (2) the controller receives traffic demand estimates from the APs for τ ms, (3) the controller retrieves station groups via station group manager module, (4) it allocates time slots for stations on a per-group basis, and (5) it sends slot allocations to the APs.

a fixed period of τ (**Step 2**). We empirically set $\tau = 2\,ms$ to keep high responsiveness and network performance. The demand estimation that arrives later than $2ms$ is ignored. This is a reasonable value in local area networks where the end-to-end delay between the controller and APs can be on the order of a few hundreds of microseconds. In our campus network, we observe round-trip times are almost always less than $1\,ms$.

The allocator then hands over the MAC addresses of the stations to station group manager. Next, the manager clusters stations into groups based on the interference types presented in a conflict graph (**Step 3**). For instance, when a wireless link from an AP to a station is in a relationship of interference with another link, two stations belong to the same station group. Transmissions to the stations should be scheduled with non-overlapped time slots. If stations do not belong to a group, they do not interfere with any other stations; such stations acquire full access to medium within that window.

The allocator sorts the station groups in a decreasing order of their total demands (*i.e.*, the sum of the traffic demands of all stations in a group) and allocates time slots to stations on a per-group basis (**Step 4**). Hence, time slots are first allocated to the group with maximum traffic demand. This allows the wireless network to maximize the total throughput. The detailed algorithm for allocating time slots to the sorted groups is given in Algorithm 1. Note that a station can be a member of multiple groups. If time slots for the station were already allocated, it is excluded from time slot allocation (at line 8 in Algorithm 1). As a final step (**Step 5**), the allocation information is disseminated to the APs which work in TDMA mode for the next two window times (*i.e.*, $40ms$ in our paper). During TDMA mode, APs stick to current allocations until a new allocation is fetched from the controller.

Station Group Manager. The controller determines the group to which a station should belong, by leveraging interference relationships among stations in a conflict graph in Section II. The manager constructs a conflict graph using an algorithm in [15] during a booting time of the controller. Note that our system does not rely on a particular conflict graph construction algorithm, and thus other techniques, such as [17], can also be used as an alternative.

Algorithm 1 Time slot allocation

```
1:  procedure ALLOCATOR( Ω )
2:      ▷ Ω: a set of sorted station groups
3:      ▷ W: no. of total slots per window
4:      if Ω is equal to φ then
5:          return
6:      G ← PickNextGroup( Ω )                ▷ G: a group
7:      for all s ∈ G do                      ▷ s: a station
8:          if IsAlreadyAllocated( s ) then
9:              Go to line 7
10:         k ← s.Demand / G.Demand × W
11:         n ← 0
12:         A ← APof( s )   ▷ A: AP that s is connected to
13:         while n ≤ k do
14:             m ← GetFirstUnallocatedSlot( G )
15:             Link( A to s ).add( m )
16:             n ← n + 1
17:     ALLOCATOR( Ω − G )                    ▷ Call recursively
```

The procedure for grouping stations is simple. (a) Given edge e_i (*i.e.*, a downlink or a link from an AP to a station) from a conflict graph, the algorithm selects another edge e_j ($i \neq j$) from the graph and checks the interference relationship between the two, based on the definitions of interferences in Section II. (b) If the links create either the HN or NHNEN interference, they are grouped together. (c) The manager chooses a next edge and repeats this process until all other edges are tested against e_i. These three steps ((a)-(c)) are executed for all edges in the conflict graph. Due to the asymmetric nature of wireless medium, the interfering station set can be a subset of another one, and we discard such a set. The resulting station groups therefore are not a subset of any other sets. However, some stations can belong to more than one groups because they can have different interference relationships with other stations. As already discussed, the controller prevents such stations from getting time slots more than once (see Algorithm 1). The complexity of our station grouping procedure is $O(n^2)$ where n is the number of wireless downlinks in the conflict graph.

The overall overhead of the controller system is low as updating the conflict graph can be done incrementally during Step 1 in Figure 3 and the allocation algorithm at Step 4 is straightforward. For EN interference, we exploit the Centaur's mechanism [6]; *i.e.*, APs use the same backoff time for the EN stations to them while keeping CSMA/CA being enabled. The controller informs the APs of a list of those EN stations.

D. CoMAC at APs

We design CoMAC that can elastically support TDMA and CSMA/CA while keeping it backward compatible with CSMA/CA. Specifically, CoMAC at APs runs atop CSMA/CA and informs CSMA/CA when and to which station the APs can transmit frames. Therefore, except for selecting a destination station, the underlying CSMA/CA is unmodified. Note that an AP enables TDMA mode only if it has at least one associated station for which some time slots are allocated.

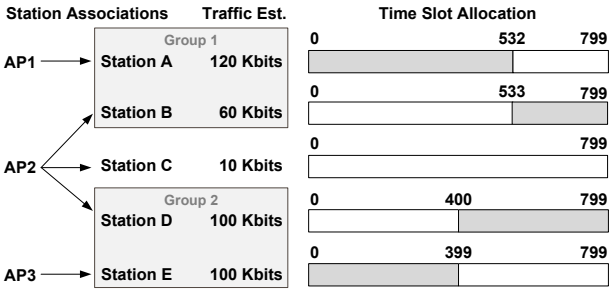


Fig. 4: An example of time slot allocation. A dark region represents allocated time slots.

Upon receiving a time slot allocation schedule from the controller, the APs begin to transmit frames as dictated in the schedule. However, there is a possibility that an AP may have a conflicting schedule for its stations or needs to share time slots with unscheduled stations that wish to receive a low volume of traffic. The AP handles such cases by selecting stations in a round-robin fashion.

To understand how the round robin strategy works, consider a scenario in Figure 4 where *Station A* is associated with *AP1*, *Stations B, C* and *D* with *AP2*, and *Station E* with *AP3*. The controller places *A* and *B* into Group 1 and *D* and *E* into Group 2 because each pair of the stations has an HN interference relationship (see Figure 2). Next, it computes a slot allocation schedule on a per-group basis (see Section III-C), which leads to an overlap of time slots for *B* and *D*. *C* has no allocated time slot because the estimated traffic volume for the station within Δ is less than $\psi = 13$ Kbits. Note that the value of ψ is empirically chosen and can be easily tuned depending on Δ. Given the scenario, since *B* and *D* own time slots within the window, and *C* does not, *AP2* can send frames to *C* in its round-robin turn between slot 0 and 799. On the other hand, *D* can occupy time slots 400-532 whereas *B* and *D* should take turns to share slots 533-799.

Once a station is selected, CoMAC calculates transmission duration for the first frame of that station in the buffer. If the transmission duration resides in the allocated time slots for the station, CoMAC dispatches the frame to CSMA/CA module for transmission. Otherwise, CoMAC reselects another station.

E. Implementation Issues

We base our evaluation on simulations, and leave a real implementation as future work. Hence, we here briefly discuss how to implement CO-FI. We envision that implementing our controller functions on an OpenFlow controller [18] is feasible. Implementing CoMAC at APs can be done through modifying the source of wireless device drivers. The MadWiFi driver is a sensible starting point, which is a widely used driver in the literature [7], [19]. MAClets [20] can also be a viable platform as it supports modification of medium access control operation. Another implementation issue is in constructing a conflict graph. We can use the active probing method in [15] or the passive method in [17]. The controller can instantly update a conflict graph using the probing method; however, it can be more complex than the passive method.

IV. EVALUATION

We now evaluate CO-FI in this section. To demonstrate its benefits, we comprehensively perform simulations using QualNet [21] and present the results. The evaluation mainly consists of two parts: i) case 1: impact of each interference type on performance in simple WLANs; and ii) case 2: performance in a realistic large-scale WLAN setup. We begin our discussion with simulation setup.

A. Basic Simulation Setup

Approaches. To evaluate the efficacy of CO-FI, we compare CO-FI with CSMA/CA and Centaur.

Traffic workloads. We first use two application-level protocols (HTTP that uses TCP and Constant Bit Rate (CBR) that uses UDP). The HTTP traffic is generated using the generation model based on the trace-driven packet analysis [22]. These traffic sources are used for investigating the impact of each interference type. For more realistic simulations, we generate TCP sessions by modeling their arrivals as a Poisson process. The exact number of TCP sessions per second is controlled by an arrival rate. Every time a TCP flow is created, its size is determined probabilistically by exploiting empirical measurement data on flow size distribution (see the CDF curve on download size in Fig. 3(a) in [11]).

Interference types. We take into account all three kinds of interferences in our simulations: HN, EN and NHNEN. For simulations in Section IV-B, we generate these interferences one by one in an isolated fashion. In contrast, all three types coexist in a realistic setting in Section IV-C.

MAC protocols. As a wired MAC protocol, we employ a Gigabit Ethernet to connect all APs and controller, and use the propagation delay of 0.5 μs. As for a wireless MAC protocol, we use IEEE 802.11n and enable Auto Rate Fallback (ARF), a rate adaptation algorithm used by CSMA/CA [9]. Finally, all the APs and stations use the same channel in the 2.4 GHz band across all simulations.

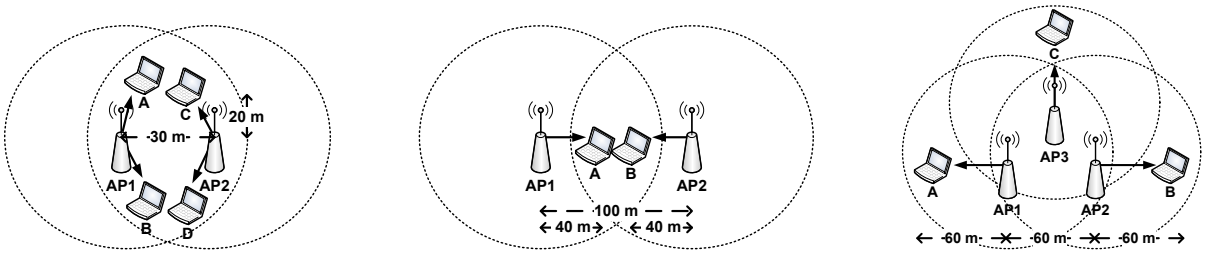
B. Influence of Each Interference Type

Configuration: We first identify what kinds of interferences our scheme can handle well. We create three controlled scenarios (NHNEN interference only, HN interference only, and EN interference only) as shown in Figure 5. For all the scenarios, the carrier-sense regions of all APs are illustrated in a reduced scale; but, all the interference relationships are preserved. The association relationships between APs and stations are presented as arrows in the figure.

Each application across all the scenarios was simulated 50 times by varying the random seed value in the QualNet simulator. The random seed affects not only the characteristics of applications, such as the number of items per Web page, but also the characteristics of a wireless environment.

Results: We evaluate the performance of our scheme in terms of throughput and end-to-end delay.

1) Throughput performance. Figure 6 shows the average throughput of HTTP traffic over 50 simulation runs. Under



(a) Non-Hidden/Non-Exposed Node (NHNEN)

(b) Hidden-Node (HN)

(c) Exposed-Node (EN)

Fig. 5: Basic simulation scenarios.

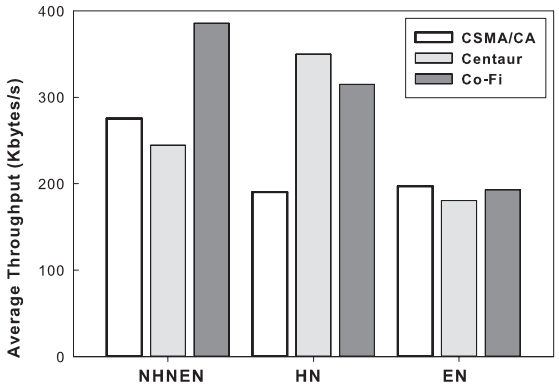


Fig. 6: Average throughput of HTTP (TCP) downlink traffic for each basic scenario.

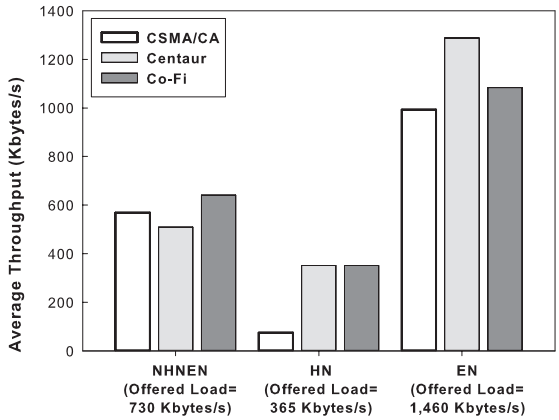


Fig. 7: Average throughput of CBR (UDP) downlink traffic for each basic scenario.

the NHNEN scenario, CO-FI achieves the highest throughput because it makes sure that frame collisions at stations occur less than other schemes. Thus, less collisions prevent ARF from slowing down a PHY transmission rate. On the other hand, other schemes face more collisions, and ARF cannot help but reduce the Tx rate, resulting in poor performance. Furthermore, Centaur obtains lower throughput than CSMA/CA. Because the Centaur controller does not forward the frames that require more time than the remaining time in a time window, frames are not often scheduled at the fringe of the current window and the next, leading to a throughput loss.

In the HN scenario, CSMA/CA achieves only about 190 KB/s as CSMA/CA is susceptible to the HN problem. In contrast, CO-FI and Centaur achieve at least 1.68× higher throughput than CSMA/CA. Between the two, Centaur slightly works better than CO-FI. As CO-FI does not schedule low volume traffic in a TDMA fashion, this may cause HN interferences to other stations. We trade this level of throughput loss for low scheduling complexity in CO-FI.

Under the EN scenario, there is no much performance difference among all three schemes. The main cause of this result is that TCP generates two-way packet streams (one for data from AP to station and the other for ACK and HTTP GET from station towards AP), and uplink streams interfere with the other downlink streams. For example, the uplink stream by station *B* and the downlink stream by *AP1* collide with each other at *AP2* due to the omnidirectional characteristic of wireless signals. Therefore, the HTTP server sends data traffic to each station sequentially; hence, there is little chance for APs to simultaneously transmit data packets in order to leverage the EN relationship.

Figure 7 shows the average throughput of CBR traffic under each controlled interference scenario. For each scenario, we use different CBR rates. Across all scenarios, both CO-FI and Centaur perform better than CSMA/CA. As opposed to the HTTP case, CO-FI and Centaur provide a clear performance benefit under the EN scenario. As already discussed, TCP generates traffic in both directions, and ACK and HTTP GET traffic can be a major source of interference. On the other hand, UDP does not face such an issue. In most cases, the performance of CO-FI is comparable to that of Centaur.

2) End-to-End delay performance. Figure 8 shows the average end-to-end delay of HTTP traffic for each scenario. CO-FI achieves the lowest end-to-end delays under the NHNEN scenario and obtains an end-to-end delay similar to that Centaur under the HN case. On the other hand, there is no visible difference in end-to-end delay in the EN case.

The simulation results with CBR traffic is somewhat different from those of HTTP traffic (see Figure 9). Before further discussion, note that in the figure, the offered CBR rates are different across interference types. The reason we vary the CBR rate is because the impacts of different interference types on delay (and throughput too) are indistinguishable among different schemes without increasing the traffic rate in the sequence of HN, NHNEN and EN. Thus, absolute delays

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

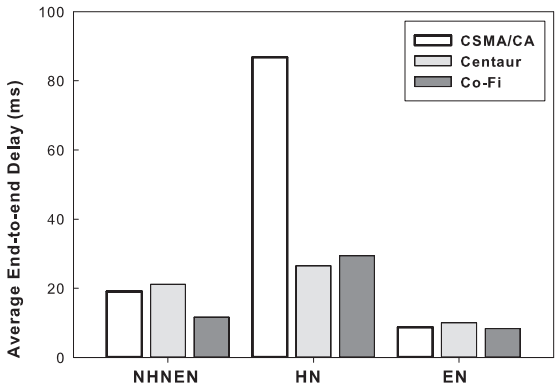


Fig. 8: Average end-to-end delay of HTTP (TCP) downlink traffic for each basic scenario.

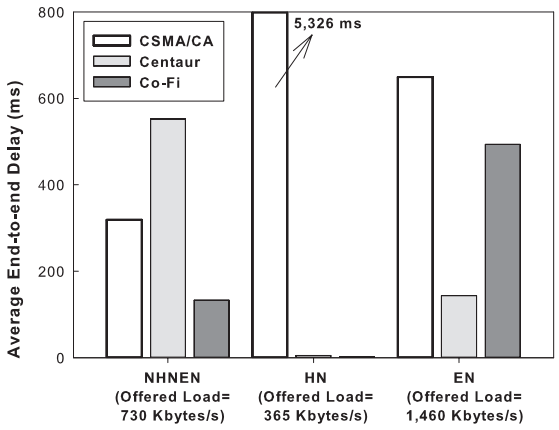


Fig. 9: Average end-to-end delay of CBR (UDP) downlink traffic for each basic scenario.

among them should not be directly compared with one another.

Under the NHNEN case, we observe that CO-FI achieves the smallest average end-to-end delay among all schemes. Centaur performs worst because it cannot properly schedule the NHNEN traffic. In the HN case, the delays of CO-FI and Centaur are negligible whereas CSMA/CA's delay is over 5 seconds. The low end-to-end delay of CO-FI and Centaur is attributed to the fact that these two schemes completely avoid the HN interference through allocating non-overlapping time slots and the offered traffic rate is relatively low. The result clearly demonstrates that a centralized traffic scheduler like CO-FI and Centaur can dramatically reduce end-to-end delays. Under the EN case, the end-to-end delay of CO-FI is almost three times higher than Centaur's.

C. Simulation under a Realistic WLAN Environment

Configuration: We create a WLAN environment in QualNet based on the WLAN environment of one of our campus buildings as illustrated in Figure 10. The created simulation environment has three floors, and all floors have the same layout and dimension (i.e., 120m length and 40m width). In addition, we place APs at the same locations where the actual APs are located in the building whilst we distribute stations randomly within the space. Because each AP uses one of

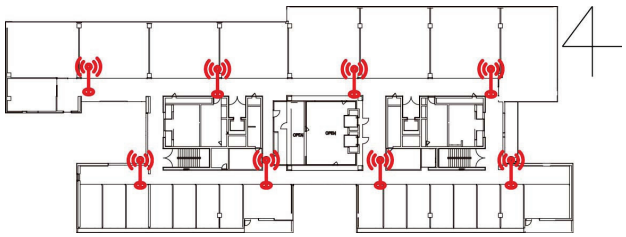


Fig. 10: Floor plan of a campus building. For simulations, stations distributed at random within the dimension of the floor plan. The characteristics of wireless medium in the simulations are set as closely to those of the building as possible. The height of a floor is around 5 meters.

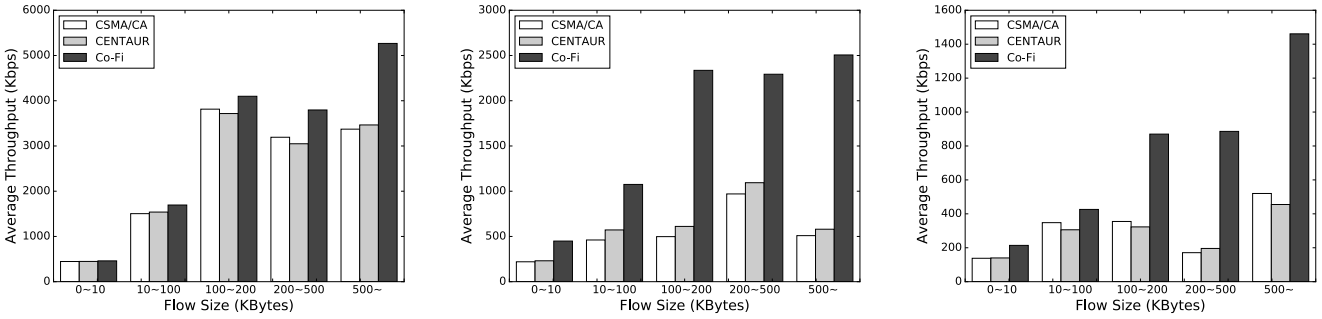
the three orthogonal channels in 2.4 GHz, we conduct our simulations under this multi-orthogonal channel environment.

We vary the number of stations (from 2 to 5 stations per AP) that are associated with each AP. Because there are 24 APs (8 APs on each floor) in total, the total number of stations varies from 48 to 120 stations, which accurately reflects WLAN environments ranging from sparse one to dense one. In addition, we use the shadowing mean of 8 dB since the wireless condition of the campus building is highly obstructed by walls and obstacles.

Results: We first evaluate average per-flow throughput. We cluster flows into groups based on their size and compute average throughput in each group. Figure 11 highlights that CO-FI outperforms Centaur and CSMA/CA across all flow size groups under the three different load conditions. As expected, there is no much gain for scheduling small flows. In contrast, for TCP flows larger than 500 KB, CO-FI achieves 3-5$\times$ higher throughput than Centaur as the load increases.

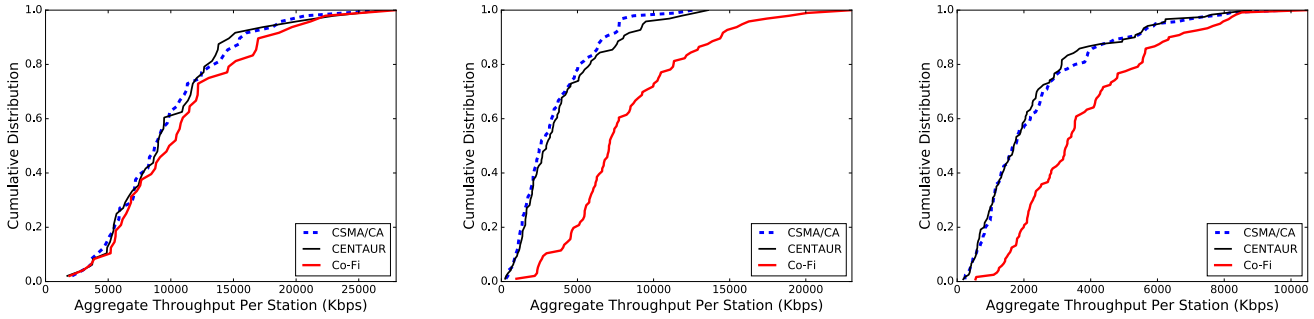
Figure 12 demonstrates cumulative distributions of averaged aggregate throughput per station. When the network is lightly loaded (i.e., 48 stations), while CO-FI works better than Centaur and CSMA/CA; however, the amount of throughput improvement is marginal. As the network becomes more crowded (96 stations), CO-FI experiences a slight throughput degradation compared to the 48 stations case, but Centaur and CSMA/CA face a significant performance drop as they cannot handle a high degree of NHNEN interference (2.01 times more number of NHNEN interferences than the 48 stations case). In case of 120 stations, CO-FI still obtains the best result, but it loses almost 47% of throughput compared to its performance in the moderately-loaded case (c.f., the median throughput in Figure 12(b) is about 7.5 Mbps and the throughput in Figure 12(c) is roughly 3.5 Mbps). Thus, the throughput difference between CO-FI and the others reduces due to the increased traffic load.

From Figures 11 and 12, we can conclude that centralized traffic scheduling mechanisms are not needed much if the network is lightly loaded. A noticeable observation is that as the network load increases, Centaur performs slightly worse than the regular CSMA/CA mechanism. This is mainly because its scheduling complexity becomes too high to bring any gain. In contrast, CO-FI sustains such a high load because the overhead



(a) 48 stations (lightly-loaded case) (b) 96 stations (moderately-loaded case) (c) 120 stations (heavily-loaded case)

Fig. 11: Average per-flow throughput depending on different flow sizes.



(a) 48 stations (lightly-loaded case) (b) 96 stations (moderately-loaded case) (c) 120 stations (heavily-loaded case)

Fig. 12: Cumulative distribution of averaged aggregate throughput per station.

of enforcing traffic schedule is distributed across APs.

The results of end-to-end delay also exhibit similar trends to those of throughput. As demonstrated in Figure 13, CO-FI reduces end-to-end delay by 10% to 90%, compared to other schemes. CO-FI always obtains the smallest end-to-end delay regardless of flow sizes and total loads. In contrast, Centaur brings marginal delay gains over CSMA/CA across the simulation cases. As the network becomes denser, the level of NHNEN interference also becomes more intensive. Centaur's lack of support for NHNEN interferences blocks further delay improvement. These results showcase that the reduced end-to-end delay of CO-FI can be particularly useful for delivering real-time (multimedia) services even in densely deployed WLAN environments.

V. RELATED WORK

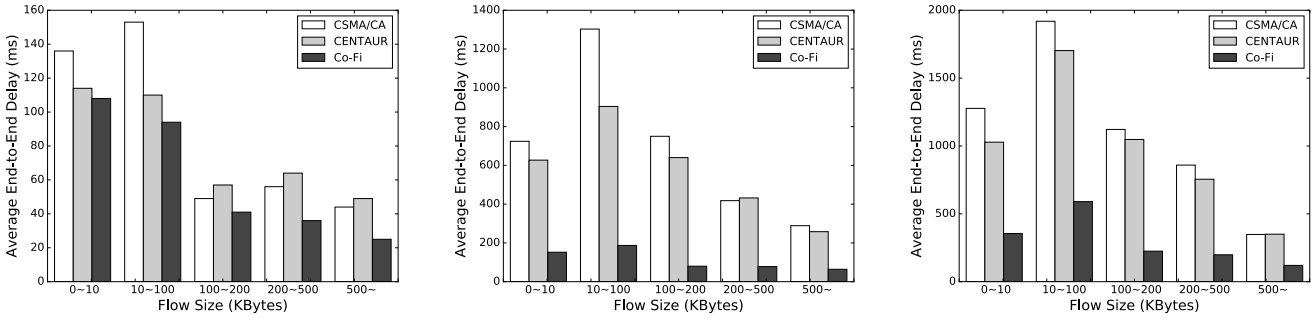
In this section, we briefly cover centralized scheduling and overlay MAC approaches that are most relevant to our work.

Centralized scheduling schemes, such as *Centaur* [6], *Shuffle* [7], and *DPS* [8], use a central controller to schedule the downlink traffic that passes through an edge router. In common, their controllers schedule every downlink frame and receives feedback from APs whenever a frame is transmitted to stations. Therefore, these schemes incur a lot of computational overhead at the controller. Further, they do not appropriately utilize the today's powerful APs because the APs strictly adhere to the transmission timing decided only by the controller. Compared to those approaches, our scheme introduces less overhead at the controller because it delegates the enforcement of time slot schedules to APs. Because Centaur is one of the most well-known centralized scheduling schemes, we did a thorough comparison of our scheme and Centaur in this work. In contrast, DOMINO [23] schedules both uplink and downlink traffic in a centralized fashion, but requires modification in PHY and MAC layers at both AP and station.

Another research topic pertaining to our scheme is overlay MAC protocols. A large body of research has focused on implementing a TDMA protocol on the 802.11-based hardware [19], [24], [25], [26]. These schemes however completely replace CSMA/CA with their own TDMA protocols. Thus, they are unfortunately incompatible with existing 802.11 devices. As the most relevant work to our hybrid MAC protocol, there exist several approaches [27], [28], [29] that implement a TDMA protocol on top of CSMA/CA without disabling it. However, these schemes have different uses of TDMA, such as fairness [27], power consumption [28], and Quality of Service (QoS) [29]. Another overlay MAC for multi-hop sensor networks switches its behavior according to the level of contention [30]. The scheme relies on a static scheduling that is determined in a distributed manner.

VI. CONCLUSION

There has been an escalated level of interference among wireless access points and stations due to dense deployment of IEEE 802.11 Wireless LANs. We introduced a coordinated Wi-Fi architecture, CO-FI, that alleviates interferences and hence boosts up wireless network performance for downlink traffic.



(a) 48 stations (lightly-loaded case) (b) 96 stations (moderately-loaded case) (c) 120 stations (heavily-loaded case)

Fig. 13: Average per-flow end-to-end delay depending on different flow sizes.

In CO-FI, a controller orchestrates access points depending on offered loads and interference types. CO-FI effectively reduces its scheduling complexity and mitigates the effect of time synchronization errors by letting a hybrid MAC protocol—*CoMAC* in the access points use CSMA/CA and TDMA protocols selectively. Our evaluation results demonstrate significantly improved throughput and end-to-end delay gain over existing approaches, especially when wireless devices are densely deployed and the networks are heavily loaded.

ACKNOWLEDGMENTS

This work was supported in part by a grant from the Royal Society.

REFERENCES

[1] A. Patro, S. Govindan, and S. Banerjee, "Observing Home Wireless Experience Through WiFi APs," in *Proceedings of ACM MobiCom*, 2013.

[2] M. A. Ergin, K. Ramachandran, and M. Gruteser, "Understanding the effect of access point density on wireless lan performance," in *Proceedings of ACM MobiCom*, 2007.

[3] V. Mhatre, K. Papagiannaki, and F. Baccelli, "Interference mitigation through power control in high density 802.11 wlans," in *Proceedings of IEEE INFOCOM*, 2007.

[4] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan, "Understanding and mitigating the impact of rf interference on 802.11 networks," in *Proceedings of ACM SIGCOMM*, 2007.

[5] Y. Lee, K. Kim, and Y. Choi, "Optimization of ap placement and channel assignment in wireless lans," in *Proceedings of IEEE LCN*, 2002.

[6] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "CENTAUR: Realizing the Full Potential of Centralized Wlans Through a Hybrid Data Path," in *Proceedings of ACM MobiCom*, 2009.

[7] J. Manweiler, N. Santhapuri, S. Sen, R. Choudhury, S. Nelakuditi, and K. Munagala, "Order matters: Transmission reordering in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 353–366, Apr. 2012.

[8] D. Zhao, M. Zhu, M. Xu, and J. Cao, "Downlink packets scheduling in enterprise wlan," in *Proceedings of IEEE WCNC*, 2013.

[9] A. Kamerman and L. Monteban, "Wavelan-ii: A high-performance wireless lan for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 118–133, 1997.

[10] M. Lacage, M. H. Manshaei, and T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," in *Proceedings of ACM MSWiM*, 2004.

[11] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A First Look at Traffic on Smartphones," in *Proceedings of ACM IMC*, 2010.

[12] A. Gupta, J. Min, and I. Rhee, "Wifox: Scaling wifi performance for large audience environments," in *Proceedings of ACM CoNEXT*, 2012.

[13] "Ericsson mobility report," http://www.ericsson.com/mobility-report, Nov. 2012.

[14] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proceedings of ACM MobiCom*, 2003.

[15] N. Ahmed and S. Keshav, "Smarta: A self-managing architecture for thin access points," in *Proceedings of ACM CoNEXT*, 2006.

[16] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *Proceedings of the 15th International Symposium on Parallel and Distributed Processing*, Apr. 2001, pp. 1965–1970.

[17] V. Shrivastava, S. Rayanchu, S. Banerjee, and K. Papagiannaki, "PIE in the Sky: Online Passive Interference Estimation for Enterprise WLANs," in *Proceedings of USENIX NSDI*, 2011.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[19] P. Djukic and P. Mohapatra, "Soft-TDMAC: A Software TDMA-Based MAC over Commodity 802.11 Hardware," in *Proceedings of IEEE INFOCOM*, 2009.

[20] G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, and I. Tinnirello, "MAClets: Active MAC Protocols over Hard-coded Devices," in *Proceedings of ACM CoNEXT*, 2012.

[21] "Qualnet – scalable network technologies," http://web.scalable-networks.com/.

[22] B. A. Mah, "An empirical model of http network traffic," in *Proceedings of IEEE INFOCOM*, 1997.

[23] W. Zhou, D. Li, K. Srinivasan, and P. Sinha, "DOMINO: Relative Scheduling in Enterprise Wireless LANs," in *Proceedings of ACM CoNEXT*, 2013.

[24] D. Koutsonikolas, T. Salonidis, H. Lundgren, P. LeGuyadec, Y. C. Hu, and I. Sheriff, "TDM MAC Protocol Design and Implementation for Wireless Mesh Networks," in *Proceedings of ACM CoNEXT*, 2008.

[25] W.-Z. Song, R. Huang, B. Shirazi, and R. LaHusen, "TreeMAC: Localized TDMA MAC Protocol for Real-time High-data-rate Sensor Networks," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 750–765, Dec. 2009.

[26] C. Li, H.-B. Li, and R. Kohno, "Reservation-based dynamic tdma protocol for medical body area networks." *IEICE Transactions*, vol. 92-B, no. 2, pp. 387–395, 2009.

[27] A. Rao and I. Stoica, "An overlay mac layer for 802.11 networks," in *Proceedings of ACM MobiSys*, 2005.

[28] J. Snow, W. chi Feng, and W. chang Feng, "Implementing a low power tdma protocol over 802.11," in *Proceedings of IEEE WCNC*, 2005.

[29] J. Lee, M. Uddin, J. Tourrilhes, S. Sen, S. Banerjee, M. Arndt, K.-H. Kim, and T. Nadeem, "meSDN: Mobile Extension of SDN," in *Proceedings of the 5th International Workshop on Mobile Cloud Computing & Services*, 2014.

[30] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-mac: A hybrid mac for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, Jun. 2008.

A Stochastic Frame Based Approach to RFID Tag Searching

Ann L. Wang[1], Muhammad Shahzad[2], Alex X. Liu[1]
[1]CSE Dept., Michigan State University; [2]CS Dept., North Carolina State University
Email:{liyanwan, alexliu}@cse.msu.edu, mshahza@ncsu.edu

***Abstract*—This paper addresses the fundamental problem of RFID tag searching: given a set of known tag IDs and a population of RFID tags with unknown IDs, where the tags may be passive or active, we want to know which tag IDs are in the tag population. RFID tag searching has many applications such as product recall, inventory balancing, and stock verification. Previous RFID tag searching protocols cannot achieve arbitrarily high accuracy and are not C1G2 compliant. In this paper, we propose a protocol called RTSP, which satisfies the four requirements of C1G2 compliance, arbitrary accuracy, privacy preserving, and multiple-reader capability. RTSP is easy to deploy because it is implemented on readers as a software module and does not require any implementation on tags. Furthermore, it does not require any modifications either to tags or to the communication protocol between tags and readers and works with the commercially available off-the-shelf RFID tags. We implemented RTSP along with the fastest tag identification protocol and compared them side-by-side. Our experimental results show that RTSP always achieves the required accuracy and is 22.73% faster than the fastest RFID identification protocol.**

I. Introduction

A. Background and Motivation

As the cost of commercial RFID tags has become negligible compared to the prices of the products to which they are attached [1], RFID systems have been increasingly used in various applications such as supply chain management [2], indoor localization [3], inventory control, and access control [4]. For example, Walmart uses RFID tags to track expensive clothing merchandize [5] and Honeywell Aerospace uses RFID tags to track its products from birth to repair and retirement [6]. An RFID system consists of tags and readers. A tag is a microchip with an integrated antenna in a compact package that has limited computing power and communication range. There are two types of tags: passive tags and active tags. Passive tags do not have their own power source, are powered up by harvesting the radio frequency energy from readers, while the active tags have their own power sources. A reader has a dedicated power source with a significant amount of computing power. RFID systems work in a query-response fashion where a reader transmits queries to a set of tags and the tags respond with their IDs over a shared wireless medium.

This paper addresses the fundamental problem of RFID tag searching: *given a set of known tag IDs and a population of RFID tags with unknown IDs, where the tags may be passive or active, we want to know which tag IDs are in the tag population,* i.e.*, search in a population of unknown tags for a set of known IDs*. RFID tag searching finds applications in product recall, inventory balancing, stock verification, and many other such settings. For product recall, if a manufacturer suspects that some of its products, which have already been distributed in different warehouses, are defective, they can use a tag searching protocol to quickly locate defective products, where the known tag IDs are defective products and the tag population are the products in a warehouse. For inventory balancing, if a large retailer, such as Amazon, wants to balance the quantity of different products among its warehouses across the country to reduce shipping time and costs, they can use a tag searching protocol to determine the quantity of any given product in each warehouse and then balance the quantity among warehouses accordingly, where the known tag IDs are the ones in inventory and the tag population are the ones in a warehouse. For stock verification, if a large retailer wants to check the quantity of each requested product sent to it in a large consignment, they can use a tag searching protocol to determine whether the consignment contains all requested products, where the known tag IDs are the ones that they are expecting and the tag population are the ones in the consignment. In this paper, we use the three terms, a tag, a tag ID, and the product that a tag is attached to, interchangeably.

B. Problem Statement

Now we formally define the tag searching problem. *Given a set A, which is a set of known tag IDs, a set B, which is a population of RFID tags with unknown IDs, a required confidence interval β, a tag searching protocol outputs $\tilde{C}$ so that $C \subseteq \tilde{C} \subseteq A$ and $|\tilde{C}| - |C| \leq \beta|C|$, where* $C = A \cap B$. Confidence interval β represents the maximum tolerable fraction of tags in A that are not in C but are declared as members of C by a tag searching protocol. A tag searching protocol should satisfy three additional requirements. First, it should comply with the EPCGlobal Class 1 Generation 2 (C1G2) RFID standard [7], which is a stable RFID standard and followed by the commercial RFID devices. Otherwise, it will be extremely difficult to be practically deployed.Second, it should preserve the privacy of the RFID tags in set B by not reading their tag IDs. Many RFID tag searching applications need to satisfy this privacy requirement. For example, if a policeman searches for some items with known tag IDs in a private house with a population of tags with unknown tag IDs, the home owner may prefer not to read the IDs of all tags in the house. Third, it should work with both a single-reader and multiple-reader environments. As the communication range

ISBN 978-3-901882-83-8

between a tag and a reader is limited, a large population of tags is often covered by multiple readers with overlapping regions.

C. Limitations of Prior Art

Previous RFID tag searching protocols (*i.e.*, [8]–[10] have two key limitations. First, they cannot achieve arbitrarily high accuracy. They are all probabilistic in nature, but none of them takes the confidence interval β as an input. Second, they do not comply with the C1G2 standard as they require the tags to receive, interpret, and act either according to pre-frame Bloom Filters or other protocol specific parameters. It is critical for RFID protocols to be compliant with the C1G2 standard because the cheap commercially available off-the-shelf (COTS) tags follow the C1G2 standard. A protocol that does not comply with the C1G2 standard will require custom tags, which will cost significantly more and have limited applications. Previous RFID identification protocols (such as TH [11], STT [12], MAS [13], and ASAP [14]) can be used to read all IDs of the tags in B and then calculate $C = A \cap B$. However, this straightforward solution has two key limitations. First, it does not preserve the privacy of the tags in B as it needs to read the IDs of all tags in B. Second, this is inefficient. We want an RFID tag searching protocol that is much faster than reading all tags in B.

D. Proposed Approach

In this paper, we propose a protocol called RFID Tag Searching Protocol (RTSP), which satisfies the following four requirement: (1) C1G2 compliance, (2) arbitrary accuracy, *i.e.*, $C \subseteq \tilde{C} \subseteq A$ and $|\tilde{C}| - |C| \leq \beta|C|$ for any required confidence interval β, (3) privacy preserving, and (4) multiple-reader capability.

To satisfy the requirement of C1G2 compliance, RTSP uses the frame slotted Aloha protocol specified in the C1G2 standard as its MAC layer communication protocol. In Aloha, the reader first tells the tags a frame size f and a random seed number R. Each tag within the transmission range of the reader then uses f, R, and its ID to select a slot in the frame by calculating a hash function $h(f, R, ID)$ whose result is uniformly distributed in $[1, f]$. Each tag has a counter initialized with the slot number that it chose to reply. After each slot, the reader first transmits an end of slot signal and then each tag decrements its counter by one. In any given slot, all the tags whose counters equal 1 respond with a random sequence called RN16. The reader uses this sequence to determine whether one or more than one tags are replying in that slot. If no tag replies in a slot, it is called an *empty slot*. If one or more tags reply in a slot, it is called a *nonempty slot*. Using 0 to denote an empty slot and 1 to denote a nonempty slot, after we execute the Aloha protocol on a population A of tags using frame size f and random seed R, we obtain a binary array of f bits, denoted as $\mathbb{S}(A, f, R)$.

To satisfy the requirement of arbitrary accuracy, RTSP executes n runs of the Aloha protocol where each run uses a different seed. For the i^{th} run with frame size f and random seed R_i, RTSP executes the Aloha protocol on both sets A and B, and thus obtains two binary arrays $\mathbb{S}(A, f, R_i)$ and $\mathbb{S}(B, f, R_i)$. Note that RTSP executes the Aloha protocol on A virtually as it knows all tag IDs in A. After n runs, for each tag ID $t \in A$, if for all $1 \leq i \leq n$, we have $\mathbb{S}(A, f, R_i)[h(f, R_i, t)] = \mathbb{S}(B, f, R_i)[h(f, R_i, t)]$, (*i.e.*, for all n runs, the two bits corresponding to tag t in both $\mathbb{S}(A, f, R_i)$ and $\mathbb{S}(B, f, R_i)$ are 1), then RTSP outputs $t \in \tilde{C}$. Clearly RTSP satisfies $C \subseteq \tilde{C} \subseteq A$. RTSP chooses a value of n so that $|\tilde{C}| - |C| \leq \beta|C|$.

To satisfy the requirement of privacy preserving, RTSP checks if a slot is empty or nonempty using the RN16 sequence and never asks tags to transmit their IDs. In C1G2, tags do not transmit their IDs unless the reader specifically asks them.

To satisfy the requirement of multi-reader capability, RTSP uses a central controller for all readers to use the same values for frame size f and seed R across all readers. The central controller uses a reader scheduling protocol [15] to ensure that two readers with overlapping regions do not transmit at the same time. When a reader transmits seed R_i in its i^{th} frame, it does not generate R_i on its own, rather, it uses the i^{th} seed R_i issued by the central controller. Thus, for a tag $t \in B$ that is covered by multiple readers, it chooses the same slot $h(f, R_i, t)$ for all readers. Once a reader completes its frame, it sends its binary array to the central controller. The controller applies the bit-wise logical OR operation on the binary arrays returned from all readers. The resulting binary array is the same as if there is one reader that covers all tags. RTSP uses this binary array to compute $\tilde{C}$.

E. Technical Challenges and Proposed Solutions

There are two key technical challenges in RTSP. The first technical challenge is to minimize tag searching time under the constraint that RTSP satisfies the required accuracy. To address this challenge, we use the accuracy requirement $|\tilde{C}| - |C| \leq \beta|C|$ to derive a *confidence condition*, which the system parameters such as frame sizes and execution rounds must satisfy. We then use the confidence condition to derive a *duration condition*, which system parameters must satisfy to minimize tag searching time. We then solve both conditions simultaneously to calculate the optimum system parameters that minimize tag searching time while achieving the required accuracy. The second technical challenge is to estimate the number of tags in set $|C|$, which is required to calculate the optimal values of system parameters. To address this challenge, RTSP counts the number of bits that are 1s in both $\mathbb{S}(A, f, R_i)$ and $\mathbb{S}(B, f, R_i)$. We call such bits dual-nonempty bits. The number of such dual-nonempty bits is a monotonically increasing function of $|C|$. By observing the number of dual-nonempty slots, RTSP estimates the value of $|C|$ while executing the Aloha protocol.

F. Advantages over Prior Art

The key novelty of this paper is in proposing a tag searching protocol that statistically guarantees to achieve any required accuracy and complies with the C1G2 standard. The key technical depth of RTSP lies in its mathematical development to guarantee any required accuracy and to minimize tag searching time. The key advantages of RTSP over prior tag

searching protocols are that RTSP can achieve arbitrarily high accuracy and RTSP complies with the C1G2 standard. RTSP is easy to deploy because it is implemented on readers as a software module and does not require any implementation on tags. Furthermore, it does not require any modifications either to tags or to the communication protocol between tags and readers and works with the commercially available off-the-shelf RFID tags. RTSP can be implemented as a software module on readers. We have extensively evaluated the performance of RTSP. Our results show that for a scenario with $|A| = 5000$, $|B| = 5000$, and $|C| = 500$, and a required confidence interval of 0.1%, RTSP takes 15 seconds to search the tags whereas the fastest prior tag identification protocol (TH [11]) takes 22 seconds.

II. Related Work

To the best of our knowledge, there are four tag searching protocols [8]–[10], [16]. Zheng and Li proposed the first RFID tag searching protocol namely CATS [8]. CATS works in two phases. In the first phase, a server first constructs a Bloom filter by applying multiple hash functions in conjunction with a random seed on each tag ID in set A. Second, an RFID reader broadcasts the Bloom filter generated by the server along with the random seed to all tags in the population B. Using the received Bloom filter of set A, each tag in B checks if it is a candidate in C. Specifically, if all bits for a tag are 1s, the tag is a candidate in C; otherwise, it must be in $B - A$. Let B' denote all these candidates. Thus, due to false positives, $C \subseteq B' \subseteq B$. Then, the tags in B' distributively constructs another Bloom filter using the Framed Slotted Aloha protocol. The reader uses this Bloom filter to exclude the IDs in $A - B$. Thus, the reader obtains the searching result $A - (A - B) = A \cap B$. Unfortunately, C1G2 compliant tags can not interpret or generate Bloom filters, which makes CATS non-compliant with the C1G2 standard.

Chen *et al.* proposed another tag searching protocol called ITSP, which is an improved version of CATS [9]. In ITSP, the reader first generates a $k = 1$ Bloom filter on set A. Then, the reader broadcasts the Bloom filter along with the parameters used for constructing the Bloom filter to all tags in B. After a tag receives the Bloom filter, it checks whether it is in the Bloom filter. If a tag is in the Bloom filter, the tag will remain active; otherwise, it will become inactive. For the active tags, they collaboratively construct another $k = 1$ Bloom filter by executing the Framed Sloted Aloha protocol. ITSP repeats the above filtering process for multiple rounds until the false positive probability is below a certain threshold. Unfortunately, C1G2 compliant tags can not interpret or generate Bloom filters, which makes ITSP non-compliant with the C1G2 standard.

Zhang *et al.* proposed another tag searching protocol called TSM [10]. TSM extends CATS for use with multiple readers. It first executes CATS using each reader and then aggregates results from all readers to identify the tags in A that are present in B. Unfortunately, due to similar reasons as for CATS, TSM is also non-compliant with the C1G2 standard. In contrast, our proposed protocol, RTSP, is C1G2 compliant.

Liu *et al.* proposed BKC to count the number of tags in A that are present in B [16]. BKC first pre-computes a frame using IDs in set A and then executes a frame on population B to determine how many times the slots that were 1 in the pre-computed frame turned out to be 1 in the executed frame. It then uses the number of such slots to obtain the estimate of the number of tags in A that are present in B. BKC falls short because it can only estimate the number of tags in A that are present in B, but it can not determine exactly which tags of A are present in B. In contrast, our proposed protocol RTSP can identify such tags.

III. System Model

A. Architecture

For searching RFID tags, RTSP uses a central controller connected with a set of readers that cover the area where the tags in set B are located. The use of a central controller ensures that all readers use consistent values of frame sizes and seeds when executing frames, which helps in efficiently aggregating and processing information returned by the readers. The readers use the standardized frame slotted Aloha protocol to communicate with tags and never ask the tags to transmit their IDs. The use of multiple readers with overlapping coverage regions introduces following two problems: (1) scheduling the readers such that no two readers with overlapping regions transmit at the same time, and (2) alleviating the effect of some tags responding to multiple readers due to overlap in the coverage region of those readers. For the first problem, the controller uses one of the several existing reader scheduling protocols [15] to avoid reader-reader collisions. For the second problem, we propose solution in Section IV-A. RTSP does not require any modifications to tags or readers. It only requires the readers to receive system parameters from the controller and communicate the responses in the frames back to the controller.

B. C1G2 Compliance

RTSP does not require any modifications to tags or readers. It only requires the readers to receive the frame size, persistence probability, and seed number from the controller and communicate the responses in the frames back to the controller. Persistence probability p is the probability with which a tag decides whether it will participate in a frame or not before selecting a slot in that frame. Later in the paper, we will show how we use p to handle frame sizes that exceed the C1G2 specified upper limit of 2^{15}. Such large frame sizes are required when the size of tag population is large and required confidence interval β is small. With the use of p, the reader reduces the number of tags that participate in each frame, which in turn reduces the optimal frame size at the expense of increased number of frames. As the C1G2 standard does not specify the use of p, COTS tags do not support it. To avoid making any modifications to tags, in RTSP, the reader implements p by announcing a frame size of f/p but terminating the frame after the first f slots and sending a command to tags to reset their counters, which can be done as per the C1G2 standard.

C. Communication Channel

We assume that the communication channel between readers and tags is reliable *i.e.*, tags correctly receives queries from the readers and the readers correctly detect transmission of RN16 sequence in a slot if one or more tags in the population transmit in that slot. If the channel is unreliable, the solution proposed in [11] can be easily adapted for use with RTSP.

D. Independence Assumption

To make the formal development tractable, we assume that instead of picking a single slot to transmit at the start of i^{th} frame of size f, a tag independently decides to transmit in each slot of the frame with probability $1/f$ regardless of its decision about previous or forthcoming slots. Vogt first used this assumption for the analysis of Aloha protocol for RFID and justified its use by recognizing that this problem belongs to a class of problems called *occupancy problem*, which deals with the allocation of balls to urns [17]. Ever since, the use of this assumption has become a norm in the formal analysis of all Aloha based RFID protocols [17]–[19].

The implication of this assumption is that a tag can end up choosing more than one slots in the same frame or even not choosing any at all, which is not in accordance with the C1G2 standard that requires a tag to pick exactly one slot in a frame. However, this assumption does not create any problems because the expected number of slots that a tag chooses in a frame is still one. The analysis with this assumption is, therefore, asymptotically the same as that without this assumption [20]. Bordenave *et al.* further explained in detail why this independence assumption in analyzing Aloha based protocols provides results just as accurate as if all the analysis was done without this assumption [20]. This independence assumption is made only to make the formal development tractable. In our simulations, tag chooses exactly one slot at the start of frame.

IV. RFID Tag Search Protocol

A. Protocol Description

To search which tags in set A are present in the population B, in RTSP, the central controller executes n Aloha frames using the RFID readers. There are five steps involved in executing each frame. First, before executing any frame i, the controller calculates the optimal values of frame size f_i, persistence probability p_i, and generates a random seed number R_i. We will derive the expressions to calculate the values of f_i and p_i in the next section. Second, as the controller knows the IDs in set A, it virtually executes the Aloha protocol on set A and obtains the binary array $\mathbb{S}(A, f_i, R_i)$. Thus, the controller knows which bits in the binary array $\mathbb{S}(B, f_i, R_i)$ resulting from executing i^{th} frame on population B should be 1 if all the tags in A were present and a single reader covered the entire population. Third, it provides each reader with the parameters f_i, p_i, and R_i and asks each of them to execute the i^{th} frame using these parameters. The motivation behind using the same values of f_i, p_i, and R_i across all readers for the i^{th} frame is to enable RTSP to work with multiple readers with overlapping regions. As all readers use the same values of f_i, p_i, and R_i in the i^{th} frame, the slot number that a particular tag chooses in the i^{th} frame of each reader covering this tag is the same *i.e.*, $h(\frac{f_i}{p_i}, R_i, ID)$ evaluated by the tag results in same value for each reader. Fourth, each reader executes the frame on its turn as per the reader scheduling protocol and sends the responses in the frame back to the controller. Fifth, after the controller has received the i^{th} frame of each reader, it applies logical OR operator on all the received i^{th} frames and obtains the resultant bit array $\mathbb{S}(B, f_i, R_i)$. This resultant bit array $\mathbb{S}(B, f_i, R_i)$ is the same as if generated by a single reader covering all the tags. After obtaining the n bit arrays, $\mathbb{S}(B, f_i, R_i)$ for $1 \leq i \leq n$, for each tag $t \in A$, if $h(\frac{f_i}{p_i}, R_i, t) \leq f_i$ the controller checks whether $\mathbb{S}(A, f_i, R_i)[h(\frac{f_i}{p_i}, R_i, t)] = \mathbb{S}(B, f_i, R_i)[h(\frac{f_i}{p_i}, R_i, t)]$ for all n frames, *i.e.*, for all n frames, whether the two bits corresponding to tag t in both $\mathbb{S}(A, f_i, R_i)$ and $\mathbb{S}(B, f_i, R_i)$ are 1s. If true, RTSP declares that the tag t is present in B. Note that RTSP can have false positives, *i.e.*, it can declare a tag in set A to be present in population B, when the tag is actually not present. RTSP does not have false negatives.

B. Estimating Number of Tags in Set C

Recall from the previous section that before executing any frame i, the controller calculates the optimal values of frame size f_i and persistence probability p_i. To calculate these optimal values for i^{th} frame, the controller needs estimate of $|C|$ at start of the i^{th} frame, which it obtains using the responses from the tag population in the previous $i-1$ frames. We represent the estimate of $|C|$ at the start of i^{th} frame by $|\tilde{C}_i|$. As the controller executes more and more frames, *i.e.*, as i increases, the estimate $|\tilde{C}_i|$ asymptotically becomes equal to $|C|$. Next, we present a method to estimate the value of $|C|$ at start of any frame i.

The intuition behind our estimation method is that as the number of tags in set C increases, the number of corresponding bits that are 1s in both $\mathbb{S}(A, f_i, R_i)$ and $\mathbb{S}(B, f_i, R_i)$ also increases. We call such bits as dual-nonempty bits. The number of dual-nonempty bits for any given frame is a function of $|C|$ and can, therefore, be used to estimate the value of $|C|$. Next, we derive an expression that relates the number of dual-nonempty bits with the value of $|C|$, *i.e.*, we derive an expression for $E[\mathcal{N}_i^{11}]$ as a function of $|C|$, where $\mathcal{N}_i^{11}$ is random variable for number of dual-nonempty bits in the pair of arrays $\mathbb{S}(A, f_i, R_i)$ and $\mathbb{S}(B, f_i, R_i)$. To derive the expression for $E[\mathcal{N}_i^{11}]$, we need the probability that any given pair of bits in the arrays $\mathbb{S}(A, f_i, R_i)$ and $\mathbb{S}(B, f_i, R_i)$ is dual-nonempty. We calculate this probability in the following lemma.

Lemma 1. *Let A be the set of IDs of tags that we want to search for in a population. Let B be the set of IDs of tags in the population in which we search for tags in set A. Let C be the set of IDs of those tags that are present in both sets A and B. Let X_{ij} be an indicator random variable for the event that the j^{th} bit in i^{th} pair of arrays is a dual-nonempty bit. For frame size f_i and persistence probability p_i, the probability distribution of X_{ij} is given by the following equation.*

$$P\{X_{ij}=1\} = 1-(1-\frac{p_i}{f_i})^{|A|}-(1-\frac{p_i}{f_i})^{|B|}+(1-\frac{p_i}{f_I})^{|A|+|B|-|C|} \quad (1)$$

Proof. Probability that any given bit j in a pair of arrays is a dual-nonempty bit can be obtained by first calculating the probability that this bit is not a dual-nonempty bit, and then subtracting it from 1. The j^{th} bit is not dual-nonempty when one of the following three cases happens.

1) None of the tags in set A select the j^{th} slot in frame *i.e.*, the j^{th} bit in $\mathbb{S}(A, f_i, R_i)$ is 0, and none of the tags in population B select the j^{th} slot in corresponding executed frame *i.e.*, the j^{th} bit in $\mathbb{S}(B, f_i, R_i)$ is 0. We represent this event by an indicator random variable Y_{00}. The probability distribution of Y_{00} is given by the following equations.

$$P\{Y_{00}=1\} = \left(1-\frac{p}{f}\right)^{|A|+|B|-|C|} \quad (2)$$

2) One or more tags in set $A-C$ select the j^{th} slot in frame *i.e.*, the j^{th} bit in $\mathbb{S}(A, f_i, R_i)$ is 1, and none of the tags in population B select the j^{th} slot in corresponding executed frame *i.e.*, the j^{th} bit in $\mathbb{S}(B, f_i, R_i)$ is 0. We represent this event by an indicator random variable Y_{10}. The probability distribution of Y_{10} is given by the following equations.

$$P\{Y_{10}=1\} = \left(1-\left(1-\frac{p}{f}\right)^{|A-C|}\right)\left(1-\frac{p}{f}\right)^{|B|} \quad (3)$$

3) None of the tags in set A select the j^{th} slot in frame *i.e.*, the j^{th} bit in $\mathbb{S}(A, f_i, R_i)$ is 0, and one or more tags in population $B-C$ select the j^{th} slot in corresponding executed frame, *i.e.*, *i.e.*, the j^{th} bit in $\mathbb{S}(B, f_i, R_i)$ is 1. We represent this event by an indicator random variable Y_{01}. The probability distribution of Y_{01} is given by the following equations.

$$P\{Y_{01}=1\} = \left(1-\left(1-\frac{p}{f}\right)^{|B-C|}\right)\left(1-\frac{p}{f}\right)^{|A|} \quad (4)$$

The distribution of X_{ij} is given by the following equation.

$$P\{X_{ij}=1\} = 1-P\{Y_{00}=1\}-P\{Y_{10}=1\}-P\{Y_{01}=1\} \quad (5)$$

Substituting the expressions for the probability distributions of Y_{00}, Y_{10}, and Y_{01} from Equations (2) to (4), respectively, into Equation (5) and simplifying, we get Equation (1). □

Following theorem derives the expression for $E[\mathcal{N}_i^{11}]$ as a function of $|C|$.

Theorem 1. *Let A be the set of IDs of tags that we want to search for in a population. Let B be the set of IDs of tags in the population in which we search for tags in set A. Let C be the set of IDs of those tags that are present in both sets A and B. Let $\mathcal{N}_i^{11}$ be the random variable for the number of dual-nonempty bits in a pair of arrays of size f_i each. When persistence probability is p_i, the expected value of $\mathcal{N}_i^{11}$ is given by the following equation.*

$$E[\mathcal{N}_i^{11}] = f_i \times \left(1-\left(1-\frac{p_i}{f_i}\right)^{|A|}-\left(1-\frac{p_i}{f_i}\right)^{|B|} + \left(1-\frac{p_i}{f_i}\right)^{|A|+|B|-|C|}\right) \quad (6)$$

Proof. It is straight forward to see that $\mathcal{N}_i^{11} = \sum_{j=1}^{f_i} X_{ij}$. As $\{X_{i1}, X_{i2}, \ldots, X_{if_i)}\}$ forms a set of identically distributed random variables, $E[\mathcal{N}_i^{11}]$ is given by

$$E[\mathcal{N}_i^{11}] = E[\sum_{j=1}^{f_i} X_{ij}] = f_i \times E[X_{ij}]$$

As expected value of an indicator random variable equals its probability of being 1, $E[X_{ij}] = P\{X_{ij}=1\}$. Substituting value of $E[X_{ij}]$ in equation above with value of $P\{X_{ij}=1\}$ from Equation (5), we get the equation for $E[\mathcal{N}_i^{11}]$. □

Fig. 1 plots $E[\mathcal{N}_i^{11}]$ as a function of $|C|$ using Equation (6). This figure is obtained using $|A| = 200$, $|B| = 300$, $f_i = 300$ and $p_i = 1$. We observe from this figure that $E[\mathcal{N}_i^{11}]$ is a monotonically increasing function of $|C|$.

To estimate the value of $|C|$, let $\tilde{\mathcal{N}}_i^{11}$ represent the observed value of number of dual-nonempty bits for i^{th} pair of bit arrays. Replacing $E[\mathcal{N}_i^{11}]$ in Equation (6) with $\tilde{\mathcal{N}}_i^{11}$ and solving for $|C|$ gives an estimate of $|C|$. Using the well known identity $(1+x)^y \approx e^{xy}$ for small x and large y, Equation (6) can be written as follows.

$$E[\mathcal{N}_i^{11}] \approx f_i \times \left(1-e^{-\frac{p_i}{f_i}|A|}-e^{-\frac{p_i}{f_i}|B|}+e^{-\frac{p_i}{f_i}(|A|+|B|-|C|)}\right)$$

Replacing $E[\mathcal{N}_i^{11}]$ in the equation above with $\tilde{\mathcal{N}}_i^{11}$ and solving for $|C|$, we get the following equation to obtain the estimate $|\tilde{C}|$ of $|C|$.

$$|\tilde{C}| \approx |A|+|B|+\frac{f_i}{p_i}\ln\left\{\frac{\tilde{\mathcal{N}}_i^{11}}{f_i}-1+e^{-\frac{p_i}{f_i}|A|}+e^{-\frac{p_i}{f_i}|B|}\right\}$$

This estimate is obtained by utilizing the information from the i^{th} frame only. While this estimate may not be accurate, if we use the information from more frames, the estimate will become more accurate. Specifically, we leverage the well known statistical result that the variance in the observed value of a random variable reduces by x times if we take the average of x observations of that random variable. Therefore, to obtain the estimate $|\tilde{C}_i|$ of $|C|$ at the start of the i^{th} frame, we obtain an estimate from each of the previous $i-1$ frames and take their average. Solving Equation (6) for $|C|$ and averaging over past $i-1$ frames, the formal expression for $|\tilde{C}_i|$ becomes

$$|\tilde{C}_i| \approx |A|+|B|+\frac{\sum_{l=1}^{i-1}\frac{f_l}{p_l}\ln\left\{\frac{\tilde{\mathcal{N}}_l^{11}}{f_l}-1+e^{-\frac{p_l}{f_l}|A|}+e^{-\frac{p_l}{f_l}|B|}\right\}}{i-1} \quad (7)$$

Note that $|B|$ can be obtained using existing RFID estimation schemes such as ART [18]. Further note that the controller obtains this estimate without executing any additional frames. It gets this estimate from the frames it was already executing to search for tags.

V. Parameter Optimization

In this section, we will derive equations that the controller uses at the start of i^{th} frame to calculate the optimal values of frame size f_i and persistence probability p_i to minimize the execution time of RTSP while ensuring that its actual confidence interval is less than the required confidence interval. At the start of i^{th} frame, the controller uses the estimate $|\tilde{C}_i|$ along with the values of $|A|$, $|B|$, and β to calculate the optimal values of f_i and p_i. Before asking the readers to execute the i^{th} frame, the controller also calculates the minimum number of frames that it should execute, represented by n_i. Recall from

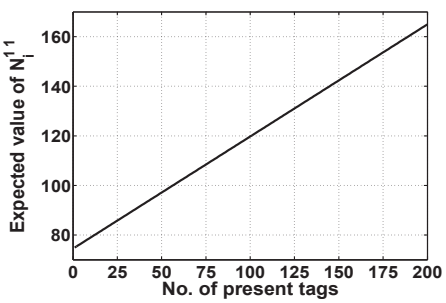


Fig. 1. $E[\mathcal{N}_i^{11}]$ vs. $|C|$

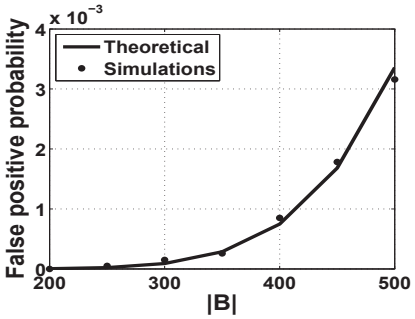


Fig. 2. Theoretical vs. experimental P_{fp}

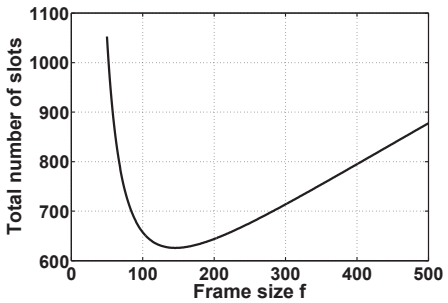


Fig. 3. Total number of slots S vs. frame size f

Section IV-B that as the number of executed frames increase, the estimate of $|C|$ becomes more accurate. Consequently, n_i, f_i, and p_i asymptotically become equal to constants n, f, and p, respectively. When the estimate of $|C|$ changes by less than 2 in 10 consecutive frames, the controller considers the estimate to be close enough to $|C|$. At this point, the controller calculates the values of n_i, f_i, and p_i one last time and puts $f = f_i$, $p = p_i$, and $n = n_i$, and uses these fixed values of f and p to execute subsequent frames until the total number of frames executed since the first frame become equal to n. For the first frame, *i.e.*, when $i = 1$, the controller uses $n_1 = \infty$, $f_1 = \max\{|A|, |B|\}$, and $p_1 = 1$. The choices of the values of n_1, f_1, and p_1 are arbitrary and do not really matter because as the controller executes more frames, number of frames, frame size, and persistence probability converge to constants n, f, and p, respectively.

In subsequent calculation of n_i, f_i, and p_i, we will drop the subscript i to make the presentation simple. Next, we first derive the expression for false positive probability *i.e.*, probability with which RTSP declares a tag in set A to be present in population B, when it actually is not. Second, using the expression for false positive probability, we derive a *confidence condition* that the values of n, f, and p must satisfy to ensure that the observed confidence interval is smaller than the required confidence interval β, *i.e.*, the requirement $|\tilde{C}| - |C| \leq \beta|C|$ is satisfied. Third, we derive a *duration condition*, which the values of f and p must satisfy to ensure that the execution time of RTSP is minimized. The controller solves these two conditions simultaneously to obtain the optimal values of n, f, and p. Last, we describe our strategy to bring the value of f within limit when the optimal frame size exceeds the C1G2 specified upper limit of 2^{15}.

A. False Positive Probability

A false positive occurs when all the bits that a particular tag in A that is not present in B selects in the n bit arrays $\mathbb{S}(A, f_i, R_i)$ for $1 \leq i \leq n$, turn out to be nonempty in corresponding bit arrays $\mathbb{S}(B, f_i, R_i)$ because some other tags in the population made those bits 1. Lemma 2 gives the expression to calculate the false positive probability.

Lemma 2. *Let B be the set of IDs of tags in the population in which we search for tags. With persistence probability p, frame size f, and number of frames n, the false positive probability, P_{fp}, is given by* $P_{fp} = \left[1 - \left(1 - \frac{p}{f}\right)^{|B|}\right]^n$.

Proof. Consider a tag t such that $t \in A \wedge t \notin B$. The probability that the bit tag t selects in $\mathbb{S}(A, f_i, R_i)$ is selected by at least one tag in population B in $\mathbb{S}(B, f_i, R_i)$ is $1-(1-\frac{p}{f})^{|B|}$. The probability that all n bits tag t selects in the n bit arrays $\mathbb{S}(A, f_i, R_i)$ for $1 \leq i \leq n$, are also selected by some other tags in population B in corresponding bit arrays $\mathbb{S}(B, f_i, R_i)$ is $[1-(1-\frac{p}{f})^{|B|}]^n$, which is the expression for P_{fp}, given in the lemma statement. □

Figure 2 shows the theoretically calculated false positive probability from equation of P_{fp} in Lemma (2) represented by the solid line and experimentally observed values of false positive probability represented by the dots. To obtain this figure, we use $f = 600$, $p = 1$, and $n = 10$. Each dot represents the false positive probability calculated from 200 runs of simulation. We observe that the theoretically calculated values match perfectly with experimentally observed values, showing that our independence assumption that we stated in Section III-D does not cause the theoretical analysis to deviate from practically observed values.

B. Confidence Condition

Theorem 2 states the confidence condition, which the values of n, f, and p must satisfy to achieve the required confidence interval β.

Theorem 2. *Let A be the set of IDs of tags that we want to search for in a population. Let B be the set of IDs of tags in the population in which we search for tags in set A. Let C be the set of IDs of those tags that are present in both sets A and B. To ensure that RTSP satisfies the requirement $|\tilde{C}| - |C| \leq \beta|C|$, the controller must use the values for number of frames n, frame size f, and persistence probability p that satisfy the confidence condition given in the following equation.*

$$n = \frac{\ln\left(\frac{\beta \times |\tilde{C}|}{|A| - |\tilde{C}|}\right)}{\ln\left(1 - (1 - \frac{p}{f})^{|B|}\right)} \tag{8}$$

Proof. Let $E[|\tilde{C}|]$ represent the number of tags that RTSP declares as belonging to set C after executing n frames of size f with persistence probability p. Replacing $|\tilde{C}|$ in $|\tilde{C}| - |C| \leq \beta|C|$ by $E[|\tilde{C}|]$, the confidence requirement is given by $E[|\tilde{C}|] - |C| \leq \beta|C|$. Next, we derive the expression for $E[|\tilde{C}|]$. Recall from Section IV-A that RTSP can have false positives, but it cannot have false negatives *i.e.*, it will always identify the tags of A present in B and in addition, it may

also declare some tags in A that are not in B to be present in B. Thus, $E[|\tilde{C}|] = |C| + (|A - C|) \times P_{fp}$. As $C \subseteq A$, thus, $E[|\tilde{C}|] = |C| + (|A| - |C|) \times P_{fp}$ Substituting this value of $E[|\tilde{C}|]$ into the confidence requirement, we get the following equation: $|C|+(|A|-|C|)\times P_{fp}-|C| \leq \beta|C|$. Substituting the value of P_{fp} from Lemma 2 into this equation and rearranging, we get $n \geq \frac{\ln\left(\frac{\beta \times |C|}{|A|-|C|}\right)}{\ln\left(1-(1-\frac{p}{f})^{|B|}\right)}$. As we do not know the exact value of $|C|$, rather we know the estimate $|\tilde{C}|$ of $|C|$, replacing $|C|$ in this equation with $|\tilde{C}|$ and using the smallest value for n allowed by the equation above to ensure that confidence requirement is always met, we get Equation (8) in theorem statement. □

C. Duration Condition

Theorem 3 states the duration condition that the values of f and p must satisfy to minimize the execution time of RTSP.

Theorem 3. *Let A be the set of IDs of tags that we want to search for in a population. Let B be the set of IDs of tags in the population in which we search for tags in set A. Let C be the set of IDs of those tags that are present in both sets A and B. To ensure that the execution time of RTSP is minimum, the controller must use the values for frame size f and persistence probability p that satisfy the duration condition given in the following equation.*

$$p \times |B| = f \times \left(1 - e^{\frac{p}{f}|B|}\right) \times \ln\left\{1 - e^{-\frac{p}{f}|B|}\right\} \quad (9)$$

Proof. Execution time is directly proportional to the total number of slots because the duration of each slot is the same, typically $300\mu s$ for Philips I-Code RFID reader [21]. Let S represent the total number of slots. Thus, $S = f \times n$. To ensure that RTSP achieves the required confidence interval, we use the value of n from Equation (8). Thus,

$$S = \frac{f \ln\left(\frac{\beta \times |\tilde{C}|}{|A|-|\tilde{C}|}\right)}{\ln\left(1 - (1 - \frac{p}{f})^{|B|}\right)} \quad (10)$$

Figure 3 plots S as a function of f using the equation above. This figure is made using $|A| = 100$, $|B| = 100$, $|\tilde{C}| = 52$, $p = 1$, and $\beta = 0.05$. We observe from this figure that S is a convex function of f. Therefore, optimum value of f exists, represented by f_{op}, that minimizes the total number of slots S. To find optimal value of f, we differentiate the equation above w.r.t f and equate the resulting expression to 0, and get the following:

$$\left[\ln\left(\frac{\beta \times |\tilde{C}|}{|A| - |\tilde{C}|}\right)\right]\left[p|B| - f\left(1 - e^{\frac{p}{f}|B|}\right)\ln\left\{1 - e^{-\frac{p}{f}|B|}\right\}\right] = 0$$

Note that $\ln\left(\frac{\beta \times |\tilde{C}|}{|A|-|\tilde{C}|}\right) \neq 0$, which means that the following must hold true: $p|B| - f\left(1 - e^{\frac{p}{f}|B|}\right)\ln\left\{1 - e^{-\frac{p}{f}|B|}\right\} = 0$ Rearranging the equation above, we get the duration condition in the theorem statement. □

The controller solves Equations (8) and (9) simultaneously using $p = 1$ and gets the optimal values of n and f represented by n_{op} and f_{op}, respectively. It calculates f_{op} numerically from Equation (9) using Brent's method. Then it puts $f = f_{\text{op}}$ and $p = 1$ in Equation (8) to calculate n_{op}. Next, we study the effect of $|A|$, $|B|$, $|C|$, and β on execution time of RTSP.

Execution Time vs. $|A|$: Intuitively, as the number of tags in A increases, the execution time of RTSP should increase because the greater number of tags in A implies the higher chances of false positives. Thus, to ensure that the number of false positives stays small enough so that the required confidence interval is achieved, RTSP executes more frames, *i.e.*, the value of n_{op} increases, which increases the overall execution time. Figure 4(a) confirms our intuition. This figure plots the expected execution time of RTSP for multiple values of $|A|$ while fixing $|B|$ at 5000 and $|C|$ at 500. We calculated the execution time as $n_{\text{op}} \times f_{\text{op}} \times T_s$, where T_s is the time of each slot and is equal to $300\mu s$ as per the specifications of Philips I-Code RFID reader [21]. We observe from Figure 4(a) that as the number of tags in A increases, the execution time of RTSP increases. The stairway behavior that RTSP shows in this and subsequent figures is due to the ceiling operation on the non-integer values of n_{op} and f_{op}.

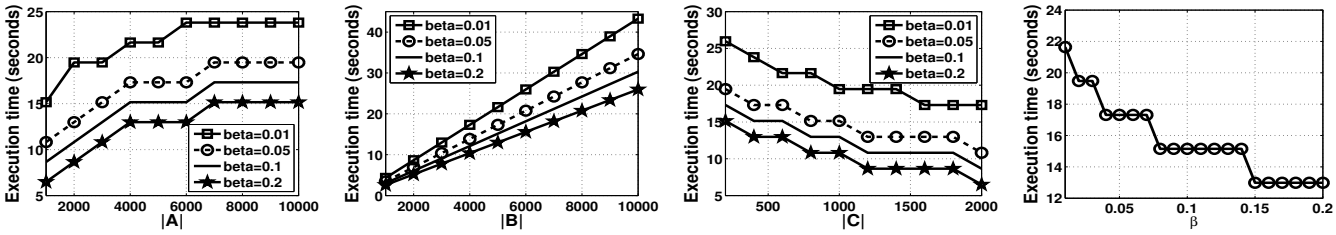
Execution Time vs. $|B|$: Intuitively, as the number of tags in B increases, the execution time of RTSP should increase because greater number of tags in B also imply higher chances of false positives. Thus, to ensure that the number of false positives stays small enough so that the required confidence interval is achieved, RTSP increases the frame size, *i.e.*, the value of f_{op} increases according to Equation (9), which increases the overall execution time. Figure 4(b) confirms our intuition. This figure plots the expected execution time of RTSP for multiple values of $|B|$ while fixing $|A|$ at 5000 and $|C|$ at 500. We observe from Figure 4(b) that as the number of tags in B increases, the execution time of RTSP increases.

Execution Time vs. $|C|$: Intuitively, as the number of tags in C increases, the execution time of RTSP should decrease because greater number of tags in C means RTSP has greater margin of error *i.e.*, $\beta|C|$. Thus, RTSP reduces the value of n_{op}, which decreases the overall execution time. Figure 4(c) confirms our intuition. This figure plots the expected execution time of RTSP for multiple values of $|C|$ while fixing $|A|$ at 5000 and $|B|$ at 5000. We observe from Figure 4(c) that as the number of tags in C increases, the execution time of RTSP decreases.

Execution Time vs. β: Intuitively, as the required confidence interval β increases, the execution time of RTSP should decrease because larger required confidence interval means RTSP has greater margin of error. Thus, RTSP reduces the values of n_{op}, which decreases the overall execution time. Figure 4(d) confirms our intuition. This figure plots the expected execution time of RTSP for different values of β while fixing $|A|$ at 5000, $|B|$ at 5000, and $|C|$ at 500. We observe from Figure 4(d) that as the required confidence interval increases, the execution time of RTSP decreases.

D. Handling Large Frame Sizes

For large populations and/or small required confidence interval, it is possible for the value of f_{op} to exceed the C1G2 specified upper limit of 2^{15}. Next, we describe how we use p



(a) Execution time vs. $|A|$ (b) Execution time vs. $|B|$ (c) Execution time vs. $|C|$ (d) Execution time vs. β

Fig. 4. Effect of $|A|$, $|B|$, $|C|$, and β on execution time of RTSP

to bring the frame size within limits. Bringing the frame size within limits comes at a cost of increased number of slots; greater than the minimum value of S that would have been achieved if the controller could use $f_{\text{op}} > 2^{15}$.

When we decrease the value of p, the number of tags that participate in a frame decreases. Therefore, the required value of f also decreases. Participation by fewer tags means that participation by the tags belonging to both the sets A and B decreases. This increases the chances that a given tag in A that is present in B will not select any slot in a given pre-computed frame, which means that chances of identifying its presence decrease. Therefore, the overall uncertainty in identifying tags in A increases. To reduce this uncertainty, the n increases when p decreases to achieve the required confidence interval.

We use these two observations to reduce the value of f whenever $f_{\text{op}} > 2^{15}$. When $f_{\text{op}} > 2^{15}$, the controller uses $f = f_{\text{max}} = 2^{15}$ in Equation (8), which leaves two unknowns, p and n, in the resulting equation. The controller solves the resulting equation simultaneously with Equation (9) to get new values of p and n. The new value of p is less than 1 and the new value of n is greater than n_{op} (we represent n with n_{op} only when we use $f = f_{\text{op}}$ to calculate it). The controller uses these new values of n and p along with $f = f_{\text{max}}$ to compute the bit array $\mathbb{S}(A, f_i, R_i)$. Although the total number of slots $S = f_{\text{max}} \times n > f_{\text{op}} \times n_{\text{op}}$, this is still the smallest under the constraints that the required confidence interval is achieved and the frame size does not exceed f_{max}.

VI. Performance Evaluation

We implemented and simulated RTSP in Matlab. We also implemented and simulated the fastest existing tag identification protocol, TH [11], to compare the execution time of RTSP with it. We choose tag ID length of 64 bits as specified in the C1G2 standard. Note that the distributions of the IDs of tags in A and B do not matter because RTSP is independent of ID distributions. Next, we first evaluate the accuracy of RTSP and then compare its execution time with the execution time of TH. All results reported in this section are obtained from averaging over 200 independent runs of RTSP.

A. Accuracy

To evaluate the accuracy of RTSP, we study its confidence interval for different values of $|A|$, $|B|$, and $|C|$.

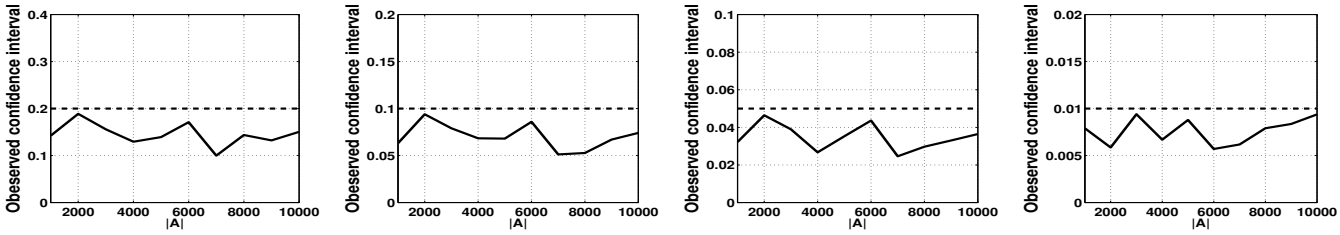
1) Observed Confidence interval vs. $|A|$: Our experimental results show that RTSP always achieves the required confidence interval regardless of the size of set A. Figures 5(a), 5(b), 5(c), and 5(d) plot the actual confidence interval RTSP achieved for different sizes of set A when the required values of confidence interval are $\beta = 0.2$, $\beta = 0.1$, $\beta = 0.05$, $\beta = 0.01$, respectively. To plot these figures, we fixed number of tags in set B at 5000 and number of tags in A that are in B, *i.e.*, number of tags in set C at 500. The dashed horizontal line in each of these figures shows the required value of confidence interval and the solid line shows the observed values of confidence interval achieved by RTSP. We observe from these figures that the observed values of confidence interval are always smaller than the required values of confidence interval.

2) Observed Confidence interval vs. $|B|$: Our experimental results show that RTSP always achieves the required confidence interval regardless of the number of tags in population B. We have not included figures due to lack of space.

3) Observed Confidence interval vs. $|C|$: Our experimental results show that RTSP always achieves the required confidence interval regardless of the number of tags in set C. Figures 6(a), 6(b), 6(c), and 6(d) plot the actual confidence interval RTSP achieved for different sizes of set C when the required values of confidence interval are $\beta = 0.2$, $\beta = 0.1$, $\beta = 0.05$, $\beta = 0.01$, respectively. To plot these figures, we fixed number of tags in sets A and B at 5000 each. Again, we observe from these figures that the solid lines are always below their corresponding dashed lines, which means that RTSP always achieves the required confidence interval.

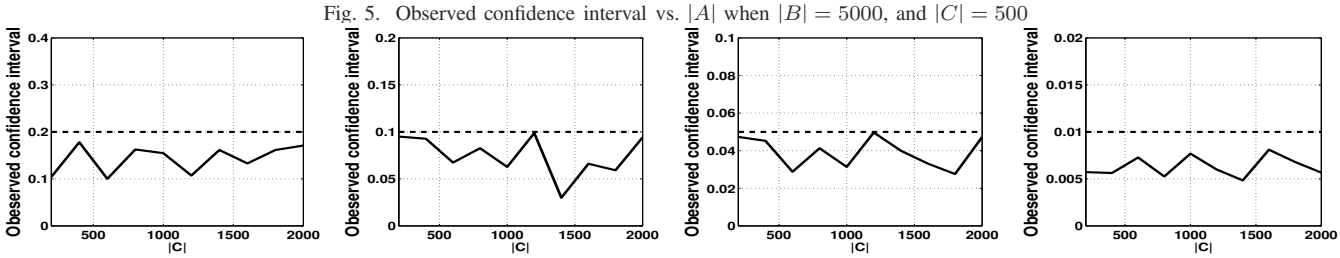
B. Execution Time

Execution time of RTSP is smaller than TH. Figure 7(a) plots the execution times of TH and RTSP vs. $|A|$ for $\beta = 0.1$, $|B| = 3000$, and $C = 500$. We observe from this figure that RTSP is up to 22.73% faster compared to TH. Similarly, Figure 7(b) plots the execution times vs. $|B|$ for $\beta = 0.1$, $|A| = 1000$, and $|C| = 500$ and Figure 7(c) plots the execution times vs. $|C|$ for $\beta = 0.1$, $|A| = 5000$, and $|B| = 5000$. Again, we observe from these figures that RTSP is always faster compared to TH. Finally, Figure 7(d) plots the execution times vs. β for $|A| = 5000$, $|B| = 5000$, and $|C| = 500$. We observe that RTSP is faster compared to TH as long as required confidence interval is > 0.01. When the required confidence interval < 0.01, TH is faster. Thus, if privacy is not a concern, a user should use TH whenever $\beta < 0.01$. If, however, privacy is a concern, the user should always use RTSP.



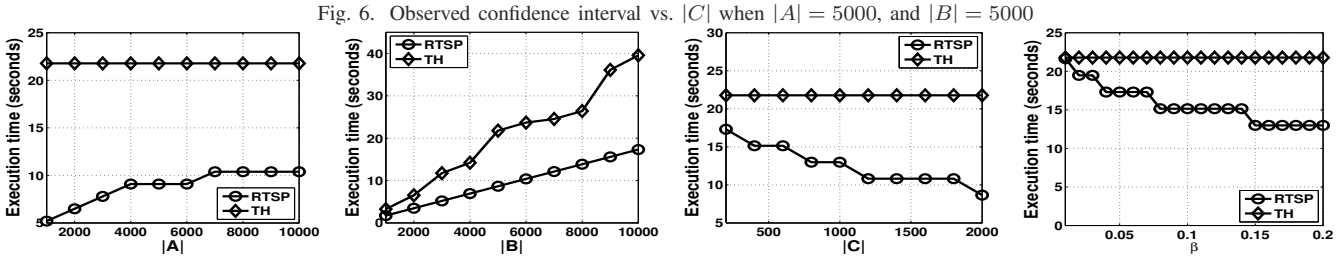
(a) required $\beta = 0.2$ (b) required $\beta = 0.1$ (c) required $\beta = 0.05$ (d) required $\beta = 0.01$

Fig. 5. Observed confidence interval vs. $|A|$ when $|B| = 5000$, and $|C| = 500$



(a) required $\beta = 0.2$ (b) required $\beta = 0.1$ (c) required $\beta = 0.05$ (d) required $\beta = 0.01$

Fig. 6. Observed confidence interval vs. $|C|$ when $|A| = 5000$, and $|B| = 5000$



(a) Execution time vs. $|A|$ (b) Execution time vs. $|B|$ (c) Execution time vs. $|C|$ (d) Execution time vs. β

Fig. 7. Comparison of execution times of RTSP and TH

VII. Conclusion

The key technical contribution of this paper is in proposing a protocol to search tags in a population of RFID tags. This paper represents the first effort on addressing this important and practical problem for C1G2 compliant RFID systems. The key technical depth of this paper is in the mathematical development of the theory that RTSP is based on. The solid theoretical underpinning ensures that RTSP always achieves the required confidence interval. We have proposed a technique to handle large frame sizes to ensure the compliance with the C1G2 standard. We have also proposed a method to implicitly estimate the number of tags in set C.We implemented RTSP and conducted side-by-side comparisons with TH, the fastest prior tag identification protocol. Our experimental results show that RTSP always achieves the required confidence interval and significantly outperforms TH in terms of search time.

References

[1] M. Roberti, "A 5-cent breakthrough," *RFID Journal*, vol. 5, no. 6, 2006.

[2] C. H. Lee and C.-W. Chung, "Efficient storage scheme and query processing for supply chain management using RFID," in *Proc. ACM Conf. on Management of data*, pp. 291–302, 2008.

[3] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: Indoor location sensing using active RFID," *Wireless networks*, vol. 10, 2004.

[4] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. Wiley, 2010.

[5] M. Roberti, "Wal-mart relaunches EPC RFID effort, starting with men's jeans and basics," *RFID Journal*, 2010.

[6] Swedberg, "Honeywell aerospace tags parts for airbus," *RFID Journal*.

[7] E. Inc, *Radio-Frequency Identity Protocols Class-1 Generation-2 Protocol for Communications at 860 MHz–960 MHz*. EPCGlobal Inc.

[8] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale RFID systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 3, pp. 924–934, 2013.

[9] M. Chen, W. Luo, Z. Mo, S. Chen, and Y. Fang, "An efficient tag search protocol in large-scale RFID systems," in *Proc. IEEE INFOCOM*, 2013.

[10] S. Zhang, X. He, H. Song, and D. Zhang, "Time efficient tag searching in multiple reader RFID systems," in *Proc. Green Computing and Communications (GreenCom)*, IEEE, 2013.

[11] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for RFID identification," in *Proc. ACM SIGMETRICS*, 2013.

[12] L. Pan and H. Wu, "Smart trend-traversal: A low delay and energy tag arbitration protocol for large RFID systems," in *Proc. INFOCOM*, 2009.

[13] V. Namboodiri and L. Gao, "Energy-aware tag anticollision protocols for RFID systems," in *Proc. 5th IEEE Int. Conf. on Pervasive Computing and Communications*, pp. 23–36, 2007.

[14] C. Qian, Y. Liu, H. Ngan, and L. M. Ni, "ASAP: Scalable identification and counting for contactless RFID systems," in *Proc. 30th IEEE Int. Conf. on Distributed Computing Systems*, pp. 52–61, 2010.

[15] S. Tang, J. Yuan, X.-Y. Li, G. Chen, Y. Liu, and J. Zhao, "Raspberry: A stable reader activation scheduling protocol in multi-reader RFID systems," in *Proc. IEEE ICNP*, 2009.

[16] X. Liu, K. Li, H. Qi, B. Xiao, and X. Xie, "Fast counting the key tags in anonymous RFID systems," in *Proc. IEEE ICNP*, 2014.

[17] H. Vogt, "Efficient object identification with passive RFID tags," *Pervasive Computing*, vol. 2414, pp. 98–113, 2002.

[18] M. Shahzad and A. X. Liu, "Every bit counts: fast and scalable RFID estimation," in *Proc. ACM MobiCom*, 2012.

[19] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for multiple RFID objects identification," *IEICE Transactions on Communications*, vol. 88, pp. 991–999, 2005.

[20] C. Bordenave, D. McDonald, and A. Proutiere, "Performance of random medium access control, an asymptotic approach," in *Proc. ACM SIGMETRICS*, 2008.

[21] P. Semiconductors, *http://www.advanide.com/datasheets/sl2ics11.pdf*.

Centralized Multi-Cell Resource and Power Allocation for Multiuser OFDMA Networks

Mohamad Yassin[‡*], Samer Lahoud[‡], Marc Ibrahim[*], Kinda Khawam[§], Dany Mezher[*], Bernard Cousin[‡]
[‡]University of Rennes 1, IRISA, Campus de Beaulieu, 35042 Rennes, France
[*]Saint Joseph University of Beirut, ESIB, Campus des Sciences et Technologies, Mar Roukoz, Lebanon
[§]University of Versailles, PRISM, 45 Avenue des Etats-Unis, 78035 Versailles, France

***Abstract*—Multiuser Orthogonal Frequency Division Multiple Access (OFDMA) networks, such as Long Term Evolution networks, use the frequency reuse-1 model to face the tremendous increase of mobile traffic demands, and to increase network capacity. However, inter-cell interference problems are generated, and they have a negative impact on cell-edge users performance. Resource and power allocation should be managed in a manner that alleviates the negative impact of inter-cell interference on system performance. In this paper, we formulate a novel centralized multi-cell resource and power allocation problem for multiuser OFDMA networks. The objective is to maximize system throughput while guaranteeing a proportional fair rate for all the users. We decompose the joint problem into two independent problems: a resource allocation problem and a power allocation problem. We prove that each of these problems is a convex optimization problem, and that their optimal solution is also an optimal solution to the original joint problem. Lagrange duality theory and subgradient projection method are used to solve the centralized power allocation problem. We study the convergence of our centralized approach, and we find out that it reduces inter-cell interference, and increases system throughput and spectral efficiency in comparison with the frequency reuse-1 model, reuse-3 model, fractional frequency reuse, and soft frequency reuse techniques.**

***Index Terms*—Convex optimization, resource and power allocation, inter-cell interference, ICIC, OFDMA.**

I. Introduction

Multiuser Orthogonal Frequency Division Multiple Access (OFDMA) networks, such as the Third Generation Partnership Project (3GPP) Long Term Evolution (LTE) [1] and LTE-Advanced (LTE-A) [2] networks, are able to avoid the negative impact of multipath fading and intra-cell interference, by virtue of the orthogonality between subcarrier frequencies. Nevertheless, Inter-Cell Interference (ICI) problems arise in dense frequency reuse networks due to simultaneous transmissions on the same frequency resources. System performance is interference-limited, since the achievable throughput is reduced due to ICI.

Fractional Frequency Reuse (FFR) [3] and Soft Frequency Reuse (SFR) [4] were introduced to avoid the harmful impact of ICI on system performance, by applying static rules on Resource Block (RB) usage and power allocation between cell zones. Heuristic Inter-Cell Interference Coordination (ICIC) techniques are proposed to achieve ICI mitigation without severe degradation of the overall system throughput. In [5], a heuristic power allocation algorithm is introduced to reduce energy consumption and to improve cell-edge UEs throughput. It has been proven that the proposed algorithm reduces power consumption without reducing the achievable throughput. Moreover, it mitigates ICI and increases the achievable throughput for cell-edge UEs.

Beside heuristic resource and power allocation algorithms [6], convex optimization is used to improve the performance of multiuser OFDMA networks, and to alleviate the negative impact of ICI on UE throughput. Resource and power allocation problems are usually formulated as nonlinear optimization problems, where the objective consists in maximizing system throughput, spectral efficiency, or energy efficiency, with constraints on the minimum throughput per UE or other Quality of Service (QoS) parameters [7].

The majority of state-of-the-art contributions formulate the resource and power allocation problem for a single cell network [8–10]. Moreover, low-complexity suboptimal algorithms are proposed to perform resource and power allocation [10]. Therefore, the optimal solution is not always guaranteed.

In this paper, we formulate the joint resource and power allocation problem for multiuser OFDMA networks, as a centralized optimization problem. We demonstrate that the original problem is separable into two independent optimization problems: a resource allocation problem and a power allocation problem. Our objective is to maximize system throughput while guaranteeing proportional fair rate among the UEs, under constraints related to resource usage, Signal-to-Interference and Noise Ratio (SINR), and power allocation. Our major contributions are summarized as follows:

- Propose an original formulation of the centralized joint resource and power allocation problem: instead of considering a single cell OFDMA network, we formulate our problem for a multi-cell OFDMA network. Moreover, ICI problems are taken into account.
- Maximize the mean rate per UE, and ensure a proportional fair rate for all the active UEs.
- Prove the convexity of our centralized problem by applying an adequate variable change.
- Decompose the joint resource and power allocation problem into two independent problems.
- Solve the centralized power allocation problem using Lagrange duality theory and subgradient projection method.
- Validate the convergence of our proposed approach and evaluate its performance in comparison with the fre-

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

quency reuse-1 model, reuse-3 model, FFR, and SFR techniques.

The remainder of this paper is organized as follows. In section II, we describe the limitations of the existing state-of-the-art approaches. In section III, system model is presented followed by our joint resource and power allocation problem formulation. The joint problem is decomposed into two independent problems in section IV: a resource allocation problem and a power allocation problem. We also demonstrate the convexity of the formulated problems. In section V, we solve both resource and power allocation problems. Then we investigate the convergence of the centralized approach in section VI, where we also provide comparisons with state-of-the-art ICIC approaches. Section VII concludes this paper and summarizes our main contributions.

II. Related Work

For a given multiuser OFDMA network, resource and power allocation problem is formulated as a centralized optimization problem. Centralized inter-cell coordination is therefore required to find the optimal solution, where the necessary information about SINR, power allocation, and resource usage are sent to a centralized coordination entity.

In [11], the multi-cell optimization problem is decomposed into two distributed optimization problems. The objective of the first problem is to minimize the transmission power allocated for cell-edge UEs, while guaranteeing a minimum throughput for each UE. RB and power are allocated to cell-edge UEs so that they satisfy their minimum required throughput. The remaining RBs and the remaining transmission power are uniformly allocated to cell-center UEs. At this stage, the second problem finds the resource allocation strategy that maximizes cell-center zone throughput. An improved version of this adaptive ICIC technique is proposed in [12], where resource allocation for cell-edge UEs is performed depending on their individual channel conditions. However, the main disadvantage of this adaptive ICIC technique and the proposed improvement is that they do not consider the impact of ICI between adjacent cells when power allocation is performed.

Resource and power allocation for a cluster of coordinated OFDMA cells are studied in [13]. Energy efficiency is maximized under constraints related to the downlink transmission power. However, noise-limited regime is considered, and ICI is neglected. Moreover, energy-efficient resource allocation for OFDMA systems is investigated in [14], where generalized and individual energy efficiencies are defined for the downlink and the uplink of the OFDMA system, respectively. Properties of the energy efficiency objective function are studied, then a low-complexity suboptimal algorithm is introduced to reduce the computational burden of the optimal solution. Subcarrier assignment is made easier using heuristic algorithms. Authors of [15] consider the joint resource allocation, power allocation, and Modulation and Coding Scheme (MCS) selection problem. The joint optimization problem is separated into resource allocation and power allocation problems, and suboptimal algorithms are proposed. Another low complexity suboptimal resource allocation algorithm is proposed in [16]. The objective consists in maximizing the achievable throughput, under constraints related to resource usage in the different cells. Cooperation between adjacent cells is needed.

The majority of state-of-the-art contributions that formulate spectral efficiency or energy efficiency problems as centralized optimization problems, neglect the impact of ICI on system performance [8–10], or introduce suboptimal approaches to solve resource and power allocation problems [17–19]. Moreover, performance comparisons are not made with other distributed heuristic ICIC algorithms with a lower complexity. In the next section, we formulate our multi-cell resource and power allocation problem that takes inter-cell interference into account.

III. System Model and Problem Formulation

A. System Model

We consider the downlink of a multiuser OFDMA system that consists of I adjacent cells and K active UEs. Let $\mathcal{I} = \{1, 2, ..., I\}$ denote the the set of cells, and $\mathcal{K} = \{1, 2, ..., K\}$ the total set of active UEs. We also define $K(i)$ as the number of UEs served by cell i. Thus, we have $\sum_{i=1}^{I} K(i) = K$. The set of available RBs in each cell is denoted by $\mathcal{N} = \{1, 2, ..., N\}$.

In OFDMA networks, system spectrum is divided into several channels, where each channel consists of a number of consecutive orthogonal OFDM subcarriers [20]. An RB is the smallest scheduling unit. It consists of 12 consecutive subcarriers in the frequency domain, and seven OFDM symbols with normal cyclic prefix in the time domain [21] (or six OFDM symbols with extended cyclic prefix). Resources are allocated to UEs each Transmit Time Interval (TTI), which is equal to 1 ms. When the frequency reuse-1 model is applied along with homogeneous power allocation, each RB is allocated the same downlink transmission power $\frac{P_{max}}{N}$, where P_{max} denotes the maximum downlink transmission power per cell. The signal to interference and noise ratio for a UE k attached to cell i and allocated RB n is given by:

$$\sigma_{k,i,n} = \frac{\pi_{i,n} G_{k,i,n}}{N_0 + \sum_{i' \neq i} \pi_{i',n} G_{k,i',n}}, \quad (1)$$

where $\pi_{i,n}$ is the downlink transmission power allocated by cell i to RB n, $G_{k,i,n}$ denotes channel gain for UE k attached to cell i and allocated RB n, and N_0 is the thermal noise power. Indexes i and i' refer to useful and interfering signals respectively. Notations, symbols, parameters, and variables used within this paper are reported in Table I.

B. Problem Formulation

1) Centralized Multi-Cell Optimization Problem: We define $\theta_{k,n}$ as the percentage of time during which UE k is associated with RB n. $\theta_{k,n}$ and $\pi_{i,n}$ are the optimization variables of the joint resource and power allocation problem. Our objective is to manage resource and power allocation in a manner that maximizes system throughput and guarantees

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

TABLE I: Sets, parameters and variables in the paper

i	Index of cell
k	Index of UE
n	Index of RB
$\mathcal{I}$	Set of cells
$\mathcal{K}$	Total set of UEs
$\mathcal{K}(i)$	Set of UEs associated to cell i
$\mathcal{N}$	Set of RBs
$\rho_{k,i,n}$	Rate of UE k associated with RB n on cell i
$\pi_{i,n}$	Transmit power of cell i on RB n
$G_{k,i,n}$	Channel gain for UE k over RB n on cell i
N_0	Thermal noise density
$\theta_{k,n}$	Percentage of time RB n is allocated to UE k
$\sigma_{k,i,n}$	SINR for UE k over RB n on cell i
P_{max}	Maximum DL transmission power per cell
π_{min}	Minimum DL transmission power per RB
$\mathcal{I}'(i)$	Set of neighboring cells for cell i

throughput fairness between the different UEs. The peak rate of UE k when associated with RB n on cell i is given by:

$$\rho_{k,i,n} = \log\left(1 + \frac{\pi_{i,n}G_{k,i,n}}{N_0 + \sum_{i' \neq i} \pi_{i',n}G_{k,i',n}}\right). \quad (2)$$

Then, the mean rate of UE k is given by:

$$\sum_{n\in\mathcal{N}} \left(\theta_{k,n}.\rho_{k,i,n}\right). \quad (3)$$

Our centralized resource and power allocation problem seeks rate maximization. We make use of the logarithmic function that is intimately associated with the concept of proportional fairness [22]. Our problem is formulated as follows:

$$\underset{\boldsymbol{\theta},\boldsymbol{\pi}}{\text{maximize}} \quad \eta = \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)} \log\left(\sum_{n\in\mathcal{N}} \theta_{k,n}.\log\left(1 + \frac{\pi_{i,n}G_{k,i,n}}{N_0 + \sum_{i'\neq i}\pi_{i',n}G_{k,i',n}}\right)\right) \quad (4a)$$

$$\text{subject to} \quad \sum_{k\in\mathcal{K}(i)} \theta_{k,n} \leq 1, \ \forall n \in \mathcal{N}, \quad (4b)$$

$$\sum_{n\in\mathcal{N}} \theta_{k,n} \leq 1, \ \forall k \in \mathcal{K}(i), \quad (4c)$$

$$\sum_{n\in\mathcal{N}} \pi_{i,n} \leq P_{max}, \ \forall i \in \mathcal{I}, \quad (4d)$$

$$\pi_{i,n} \geq \pi_{min}, \ \forall i \in \mathcal{I}, \forall n \in \mathcal{N}, \quad (4e)$$

$$0 \leq \theta_{k,n} \leq 1, \ \forall k \in \mathcal{K}(i), \forall n \in \mathcal{N}. \quad (4f)$$

The objective function η ensures a proportional fair rate for all UEs in the network. Constraints (4b) ensure that an RB is used at most 100% of the time, and constraints (4c) ensure that a UE shares its time on the available RBs. Constraints (4d) guarantee that the total downlink transmission power allocated to the available RBs does not exceed the maximum transmission power P_{max} for each cell i, and constraints (4e) represent the minimum power constraint of the transmit power allocated to each RB. $\theta_{k,n}, \forall k \in \mathcal{K}, \forall n \in \mathcal{N}$, and $\pi_{i,n}, \forall i \in \mathcal{I}, \forall n \in \mathcal{N}$ are the optimization variables of the joint resource and power allocation problem.

2) Upper Bound of the Objective Functions Difference: In order to reduce the complexity of the joint resource and power allocation problem (4), we prove that this problem is separable into two independent problems: a resource allocation problem and a power allocation problem. Given Jensen's inequality and the concavity of the log function, we have:

$$\log\left(\frac{\sum_{n\in\mathcal{N}} \theta_{k,n}.\rho_{k,i,n}}{|\mathcal{N}|}\right) \geq \frac{\sum_{n\in\mathcal{N}} \log\left(\theta_{k,n}.\rho_{k,i,n}\right)}{|\mathcal{N}|} \quad (5a)$$

$$\Rightarrow \log\left(\sum_{n\in\mathcal{N}} \theta_{k,n}.\rho_{k,i,n}\right) \geq \frac{\sum_{n\in\mathcal{N}} \log\left(\theta_{k,n}.\rho_{k,i,n}\right)}{|\mathcal{N}|} + \log\left(|\mathcal{N}|\right), \quad (5b)$$

Since $\frac{1}{|\mathcal{N}|}$ and $|\mathcal{K}|.\log\left(|\mathcal{N}|\right)$ are constant terms, maximizing the objective function of problem (4) is achieved by maximizing the following term:

$$\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}} \left(\log\left(\theta_{k,n}\right) + \log\left(\rho_{k,i,n}\right)\right). \quad (6)$$

IV. Problem Decomposition

We tackle ICIC as an optimization problem, where we intend to maximize the mean rate of UEs in a multiuser OFDMA system. We consider a system of I cells, having $K(i)$ UEs per cell i. According to (6), and due to the absence of binding constraints, the optimization problem (4) is linearly separable into two independent problems: a power allocation problem and a resource allocation problem.

A. Centralized Multi-Cell Power Allocation Problem

In the first problem, the optimization variable $\boldsymbol{\pi}$ is considered, and the problem is formulated as follows:

$$\underset{\boldsymbol{\pi}}{\text{maximize}} \quad \eta_1 = \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}} \log\left(\log\left(1 + \frac{\pi_{i,n}G_{k,i,n}}{N_0 + \sum_{i'\neq i}\pi_{i',n}G_{k,i',n}}\right)\right) \quad (7a)$$

$$\text{subject to} \quad \sum_{n\in\mathcal{N}} \pi_{i,n} \leq P_{max}, \ \forall i \in \mathcal{I}, \quad (7b)$$

$$\pi_{i,n} \geq \pi_{min}, \ \forall i \in \mathcal{I}, \forall n \in \mathcal{N}. \quad (7c)$$

Problem (7) consists in finding the optimal power allocation. In the following, we introduce a variable change that allows to formulate problem (7) as a convex optimization problem.

The power allocation problem (7) can be written as follows:

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

$$\underset{\boldsymbol{\rho}}{\text{maximize}} \quad \eta_1 = \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}} \log\left(\rho_{k,i,n}\right) \tag{8a}$$

$$\text{subject to} \quad \rho_{k,i,n} \leq \log\left(1+\frac{\pi_{i,n}G_{k,i,n}}{N_0+\sum_{i'\neq i}\pi_{i',n}G_{k,i',n}}\right), \quad \forall i\in\mathcal{I},\forall k\in\mathcal{K}(i),\forall n\in\mathcal{N}, \tag{8b}$$

$$\sum_{n\in\mathcal{N}}\pi_{i,n} \leq P_{max},\ \forall i\in\mathcal{I}, \tag{8c}$$

$$\pi_{i,n} \geq \pi_{min},\ \forall i\in\mathcal{I},\forall n\in\mathcal{N}. \tag{8d}$$

Let us consider the following variable change:

$$\widehat{\rho}_{k,i,n} = \log\left(\exp\left(\rho_{k,i,n}\right)-1\right),\ \forall\, i\in\mathcal{I},\forall\, k\in\mathcal{K}(i),\forall n\ \in\mathcal{N}, \tag{9a}$$

$$\widehat{\pi}_{i,n} = \log(\pi_{i,n}),\ \forall\, i\in\mathcal{I},\forall\, n\in\mathcal{N}. \tag{9b}$$

To show that the optimization problem (8) is a convex optimization problem, we need to show that the objective function is concave and the inequality constraint functions define a convex set. After applying the variable change on UE peak rate constraints (8b), these constraints can be written as follows:

$$\log(\exp\left(\widehat{\rho}_{k.i.n}-\widehat{\pi}_{i,n}\right)\frac{N_0}{G_{k,i,n}}+ \sum_{i'\neq i}\exp\left(\widehat{\rho}_{k.i.n}+\widehat{\pi}_{i',n}-\widehat{\pi}_{i,n}\right)\frac{G_{k,i',n}}{G_{k,i,n}}) \leq 0,$$

which are the logarithmic of the sum of exponential functions. Therefore, they are convex functions [23]. When we apply the variable change on power constraints (8c), we get the following:

$$\sum_{n\in\mathcal{N}}\pi_{i,n} \leq P_{max},\ \forall i\in\mathcal{I}$$

$$\Rightarrow \log\left(\sum_{n\in\mathcal{N}}\exp\left(\widehat{\pi}_{i,n}\right)\right)-\log\left(P_{max}\right)\leq 0,\ \forall i\in\mathcal{I}.$$

Since $\log(\sum\exp)$ is convex [23], the constraints at hand are therefore convex. Using the variable change, the power allocation problem (8) can be written as follows:

$$\underset{\widehat{\boldsymbol{\rho}}}{\text{maximize}} \quad \eta_1 = \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}} \log\left(\log\left(\exp\left(\widehat{\rho}_{k,i,n}\right)+1\right)\right) \tag{10a}$$

$$\text{subject to} \quad \log(\exp\left(\widehat{\rho}_{k.i.n}-\widehat{\pi}_{i,n}\right)\frac{N_0}{G_{k,i,n}}+ \sum_{i'\neq i}\exp\left(\widehat{\rho}_{k.i.n}+\widehat{\pi}_{i',n}-\widehat{\pi}_{i,n}\right)\frac{G_{k,i',n}}{G_{k,i,n}}) \leq 0, \quad \forall i\in\mathcal{I},\forall k\in\mathcal{K}(i),\forall n\in\mathcal{N}, \tag{10b}$$

$$\log\left(\sum_{n\in\mathcal{N}}\exp\left(\widehat{\pi}_{i,n}\right)\right)-\log\left(P_{max}\right)\leq 0,\ \forall i\in\mathcal{I}, \tag{10c}$$

$$\widehat{\pi}_{i,n} \geq \log\left(\pi_{min}\right),\ \forall i\in\mathcal{I},\forall n\in\mathcal{N}. \tag{10d}$$

The objective function of problem (10) is concave in $\widehat{\boldsymbol{\rho}}$, and constraints (10b), (10c), and (10d) are convex functions. Thus, the power allocation problem is a convex optimization problem.

B. Centralized Resource Allocation Problem

The optimization variable $\boldsymbol{\theta}$ is considered in the second optimization problem that is given in the following:

$$\underset{\boldsymbol{\theta}}{\text{maximize}} \quad \eta_2 = \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}} \log\left(\theta_{k,n}\right) \tag{11a}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{K}(i)}\theta_{k,n} \leq 1,\ \forall n\in\mathcal{N}, \tag{11b}$$

$$\sum_{n\in\mathcal{N}}\theta_{k,n} \leq 1,\ \forall k\in\mathcal{K}(i), \tag{11c}$$

$$0 \leq \theta_{k,n} \leq 1,\ \forall k\in\mathcal{K}(i),\forall n\in\mathcal{N}. \tag{11d}$$

As demonstrated for the power allocation problem (7), we prove that problem (11) is indeed a convex optimization problem in $\boldsymbol{\theta}$. The objective function (11a) of the resource allocation problem (11) is concave in $\boldsymbol{\theta}$, since the log function is concave for $\boldsymbol{\theta}\in\,]0;1]$. Moreover, constraints (11b), (11c), and (11d) are linear and separable constraints. Hence, the resource allocation problem (11) is a convex optimization problem, and it is separable into $\mathcal{I}$ subproblems. For each cell i, the ith optimization problem is written as follows:

$$\underset{\boldsymbol{\theta}}{\text{maximize}} \quad \left(\eta_2\right)_i = \sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}} \log\left(\theta_{k,n}\right) \tag{12a}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{K}(i)}\theta_{k,n} \leq 1,\ \forall n\in\mathcal{N}, \tag{12b}$$

$$\sum_{n\in\mathcal{N}}\theta_{k,n} \leq 1,\ \forall k\in\mathcal{K}(i), \tag{12c}$$

$$0 \leq \theta_{k,n} \leq 1,\ \forall k\in\mathcal{K}(i),\forall n\in\mathcal{N}. \tag{12d}$$

V. Centralized Multi-Cell Resource and Power Allocation

As proven in the previous section, the joint resource and power allocation problem (4) is separable into two independent convex optimization problems: a power allocation problem, and a resource allocation problem. In this section, we solve the resource and power allocation problems using Lagrange duality theory and subgradient projection method.

A. Solving the Centralized Power Allocation Problem

1) Lagrange-Based Method: Since the power allocation problem (10) is a convex optimization problem, we can make use of Lagrange duality properties, which also lead to decomposability structures [24]. Lagrange duality theory links the original problem, or *primal problem*, with a dual maximization problem. The primal problem (10) is relaxed by transferring the constraints to the objective in the form of weighted sum. The Lagrangian is formed by relaxing the

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

coupling constraints (10b) and (10c) in problem (10):

$$L\left(\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}},\boldsymbol{\lambda},\boldsymbol{\nu}\right)=\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}}\log\left(\log\left(\exp\left(\widehat{\rho}_{k,i,n}\right)+1\right)\right)$$
$$-\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}(i)}\sum_{n\in\mathcal{N}}\lambda_{k,i,n}(\log(\exp(\widehat{\rho}_{k.i.n}-\widehat{\pi}_{i,n})\frac{N_0}{G_{k,i,n}}$$
$$+\sum_{\substack{i'\in\mathcal{N}\\ i'\neq i}}\exp(\widehat{\rho}_{k.i.n}+\widehat{\pi}_{i',n}-\widehat{\pi}_{i,n})\frac{G_{k,i',n}}{G_{k,i,n}}))$$
$$-\sum_{i\in\mathcal{I}}\nu_i\left(\log\left(\sum_{n\in\mathcal{N}}\exp\left(\widehat{\pi}_{i,n}\right)\right)-\log\left(P_{max}\right)\right). \tag{13}$$

The optimization variables $\widehat{\boldsymbol{\rho}}$ and $\widehat{\boldsymbol{\pi}}$ are called the primal variables. $\lambda_{k,i,n}$ and ν_i are the *Lagrange multipliers* or *prices* associated with the (k,i,n)th inequality constraint (10b) and with the ith inequality constraint (10c), respectively. $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are also termed the *dual variables*.

After relaxing the coupling constraints, the optimization problem separates into two levels of optimization: lower level and higher level. At the lower level, $L(\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}},\boldsymbol{\lambda},\boldsymbol{\nu})$ is the objective function to be maximized. $\widehat{\rho}_{k,i,n}$ and $\widehat{\pi}_{i,n}$ are the optimization variables to be found, and the primal problem is given by:

$$\underset{\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}}}{\text{maximize}} \quad L\left(\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}},\boldsymbol{\lambda},\boldsymbol{\nu}\right) \tag{14a}$$
$$\text{subject to} \quad \widehat{\pi}_{i,n}\geq\log(\pi_{min}),\ \forall i\in\mathcal{I},\forall n\in\mathcal{N}. \tag{14b}$$

In order to solve the primal optimization problem (14), we use the subgradient projection method. It starts with some initial feasible values of $\widehat{\rho}_{k,i,n}$ and $\widehat{\pi}_{i,n}$ that satisfy the constraints (14b). Then, the next iteration is generated by taking a step along the subgradient direction of $\widehat{\rho}_{k,i,n}$ and $\widehat{\pi}_{i,n}$. For the primal optimization variables, iterations of the subgradient projection are given by:

$$\widehat{\rho}_{k,i,n}(t+1)=\widehat{\rho}_{k,i,n}(t)+\delta(t)\times\frac{\partial L}{\partial\widehat{\rho}_{k,i,n}},$$
$$\forall k\in\mathcal{K}(i),\forall i\in\mathcal{I},\forall n\in\mathcal{N}, \tag{15a}$$
$$\widehat{\pi}_{i,n}(t+1)=\widehat{\pi}_{i,n}(t)+\delta(t)\times\frac{\partial L}{\partial\widehat{\pi}_{i,n}},\forall i\in\mathcal{I},\forall n\in\mathcal{N}. \tag{15b}$$

The scalar $\delta(t)$ is a step size that guarantees the convergence of the optimization problem (14). At the higher level, we have the master dual problem in charge of updating the dual variables $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ by solving the dual problem:

$$\underset{\boldsymbol{\lambda},\boldsymbol{\nu}}{\text{minimize}} \quad \max_{\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}}}\left(L\left(\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}},\boldsymbol{\lambda},\boldsymbol{\nu}\right)\right) \tag{16a}$$
$$\text{subject to} \quad \boldsymbol{\lambda}\geq 0, \tag{16b}$$
$$\boldsymbol{\nu}\geq 0. \tag{16c}$$

The dual function $g\left(\boldsymbol{\lambda},\boldsymbol{\nu}\right)=\max_{\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}}}\left(L\left(\widehat{\boldsymbol{\rho}},\widehat{\boldsymbol{\pi}},\boldsymbol{\lambda},\boldsymbol{\nu}\right)\right)$ is differentiable. Thus, the master dual problem (16) can be solved using the following gradient method:

$$\lambda_{k,i,n}(t+1)=\lambda_{k,i,n}(t)+\delta(t)\times\frac{\partial L}{\partial\lambda_{k,i,n}},$$
$$\forall k\in\mathcal{K}(i),\forall i\in\mathcal{I},\forall n\in\mathcal{N}, \tag{17a}$$
$$\nu_i(t+1)=\nu_i(t)+\delta(t)\times\frac{\partial L}{\partial\nu_i},\forall i\in\mathcal{I},\forall n\in\mathcal{N}, \tag{17b}$$

where t is the iteration index, and $\delta(t)$ is the step size at iteration t. Appropriate choice of the step size [25] leads to convergence of the dual algorithm. $\widehat{\pi}^{\star}_{i,n}$ and $\widehat{\rho}^{\star}_{k,i,n}$ denote the solution to the primal optimization problem (14). When $t\rightarrow\infty$ the dual variables $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\nu}(t)$ converge to the dual optimal $\boldsymbol{\lambda}^*$ and $\boldsymbol{\nu}^*$, respectively. The difference between the optimal primal objective and the optimal dual objective, called *duality gap*, reduces to zero at optimality, since the problem (10) is convex and the KKT conditions are satisfied. We define $\Delta\widehat{\boldsymbol{\rho}},\Delta\widehat{\boldsymbol{\pi}},\Delta\boldsymbol{\lambda}$, and $\Delta\boldsymbol{\nu}$ as the differences between the optimization variables obtained at the current iteration and their values at the previous iteration. They are given by:

$$\Delta\widehat{\boldsymbol{\rho}}(t+1)=\|\widehat{\boldsymbol{\rho}}(t+1)-\widehat{\boldsymbol{\rho}}(t)\|, \tag{18a}$$
$$\Delta\widehat{\boldsymbol{\pi}}(t+1)=\|\widehat{\boldsymbol{\pi}}(t+1)-\widehat{\boldsymbol{\pi}}(t)\|, \tag{18b}$$
$$\Delta\boldsymbol{\lambda}(t+1)=\|\boldsymbol{\lambda}(t+1)-\boldsymbol{\lambda}(t)\|, \tag{18c}$$
$$\Delta\boldsymbol{\nu}(t+1)=\|\boldsymbol{\nu}(t+1)-\boldsymbol{\nu}(t)\|. \tag{18d}$$

2) Iterative Power Allocation Algorithm: The procedure for solving the centralized power allocation problem is described in Algorithm 1. Initially, the primal optimization variables $\widehat{\rho}_{k,i,n}$ and $\widehat{\pi}_{i,n}$ as well as the dual variables $\lambda_{k,i,n}$ and ν_i start with some initial feasible values. t, t_{primal}, and t_{dual} denote the number of rounds required for the centralized power allocation problem to converge, the number of iterations for the primal problem, and the number of iterations for the dual problem, respectively. At each round t, we start by updating the primal optimization variables, using the PRIMALPROBLEM function given in Algorithm 2. The solution to the primal optimization problem at the current round t is denoted by $\widehat{\pi}^{\star}_{i,n}(t+1)$ and $\widehat{\rho}^{\star}_{k,i,n}(t+1)$. The PRIMALPROBLEM function updates $\widehat{\pi}_{i,n}(t_{primal}+1)$ and $\widehat{\rho}_{k,i,n}(t_{primal}+1)$, and increments t_{primal} until $\Delta\widehat{\boldsymbol{\pi}}(t_{primal}+1)$ and $\Delta\widehat{\boldsymbol{\rho}}(t_{primal}+1)$ become less than ϵ.

Then, the solution to the dual optimization problem at the current round t, denoted by $\nu^{\star}_i(t+1)$ and $\lambda^{\star}_{k,i,n}(t+1)$ is calculated using the DUALPROBLEM function given in Algorithm 3. ν_i and $\lambda_{k,i,n}$ are updated using the primal solution $\widehat{\pi}^{\star}_{i,n}(t+1)$ and $\widehat{\rho}^{\star}_{k,i,n}(t+1)$, until $\Delta\boldsymbol{\nu}(t_{dual}+1)$ and $\Delta\boldsymbol{\lambda}(t_{dual}+1)$ become less than ϵ. An additional round of calculations is performed, and t is incremented as long as $\Delta\widehat{\boldsymbol{\pi}}^{\star}(t+1)$ or $\Delta\widehat{\boldsymbol{\pi}}^{\star}(t+1)$ or $\Delta\boldsymbol{\nu}^{\star}(t+1)$ or $\Delta\boldsymbol{\lambda}^{\star}(t+1)$ is greater than ϵ. Otherwise, the current solution is the optimal solution to the centralized power allocation problem.

B. Solving the Resource Allocation Problem

In this subsection, we search for the optimal solution to the resource allocation problem (12). For each cell i, the problem (12) is a convex optimization problem.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Algorithm 1 Dual algorithm for centralized power allocation

1: Parameters: $L(\widehat{\boldsymbol{\rho}}, \widehat{\boldsymbol{\pi}}, \boldsymbol{\lambda}, \boldsymbol{\nu})$, P_{max}, and π_{min}.
2: Initialization: set $t = t_{primal} = t_{dual} = 0$, and $\pi_{i,n} \geq \pi_{min}, \forall i \in \mathcal{I}, \forall n \in \mathcal{N}$, such as $\sum_{n \in \mathbb{N}} \pi_{i,n} \leq P_{max}, \forall i \in \mathcal{I}$. Calculate $\widehat{\pi}_{i,n}(0)$ and $\widehat{\rho}_{k,i,n}(0)$ accordingly.
3: Set $\lambda_{k,i,n}(0)$ and $\nu_i(0)$ equal to some non negative value.
4: $(\widehat{\boldsymbol{\pi}}^{\star}(t+1), \widehat{\boldsymbol{\rho}}^{\star}(t+1)) \leftarrow$ PRIMALPROBLEM$(\boldsymbol{\nu}^{\star}(t), \boldsymbol{\lambda}^{\star}(t))$
5: $(\boldsymbol{\nu}^{\star}(t+1), \boldsymbol{\lambda}^{\star}(t+1)) \leftarrow$ DUALPROBLEM$(\widehat{\boldsymbol{\pi}}^{\star}(t+1), \widehat{\boldsymbol{\rho}}^{\star}(t+1))$
6: **if** $(\Delta\widehat{\boldsymbol{\pi}}^{\star}(t+1) > \epsilon)$ **or** $(\Delta\widehat{\boldsymbol{\rho}}^{\star}(t+1) > \epsilon)$ **or** $(\Delta\boldsymbol{\nu}^{\star}(t+1) > \epsilon)$ **or** $(\Delta\boldsymbol{\lambda}^{\star}(t+1) > \epsilon)$ **then**
7: $t \leftarrow t+1$
8: **go to** 4
9: **end if**

Algorithm 2 Primal problem function

1: **function** PRIMALPROBLEM$(\boldsymbol{\nu}^{\star}(t), \boldsymbol{\lambda}^{\star}(t))$
2: **for** $i = 1$ to $|\mathcal{I}|$ **do**
3: **for** $n = 1$ to $|\mathcal{N}|$ **do**
4: $\widehat{\pi}_{i,n}(t_{primal}+1) \leftarrow \max\left(\log(\pi_{min}); \widehat{\pi}_{i,n}(t_{primal}) + \delta(t) \times \frac{\partial L}{\partial \widehat{\pi}_{i,n}}\right)$
5: **for** $k = 1$ to $|\mathcal{K}(i)|$ **do**
6: $\widehat{\rho}_{k,i,n}(t_{primal}+1) \leftarrow \widehat{\rho}_{k,i,n}(t_{primal}) + \delta(t) \times \frac{\partial L}{\partial \widehat{\rho}_{k,i,n}}$
7: **end for**
8: **end for**
9: **end for**
10: **if** $(\Delta\widehat{\boldsymbol{\pi}}(t_{primal}+1) > \epsilon)$ **or** $(\Delta\widehat{\boldsymbol{\rho}}(t_{primal}+1) > \epsilon)$ **then**
11: $t_{primal} \leftarrow t_{primal}+1$
12: **go to** 2
13: **end if**
14: **return** $\widehat{\boldsymbol{\pi}}(t_{primal}+1), \widehat{\boldsymbol{\rho}}(t_{primal}+1)$
15: **end function**

Algorithm 3 Dual problem function

1: **function** DUALPROBLEM$(\widehat{\boldsymbol{\pi}}^{\star}(t+1), \widehat{\boldsymbol{\rho}}^{\star}(t+1))$
2: **for** $i = 1$ to $|\mathcal{I}|$ **do**
3: $\nu_i(t_{dual}+1) \leftarrow \max(0; \nu_i(t_{dual}) + \delta(t) \times \frac{\partial L}{\partial \nu_i})$
4: **for** $n = 1$ to $|\mathcal{N}|$ **do**
5: **for** $k = 1$ to $|\mathcal{K}(i)|$ **do**
6: $\lambda_{k,i,n}(t_{dual}+1) \leftarrow \max(0; \lambda_{k,i,n}(t_{dual}) + \delta(t) \times \frac{\partial L}{\partial \lambda_{k,i,n}})$
7: **end for**
8: **end for**
9: **end for**
10: **if** $(\Delta\boldsymbol{\nu}(t_{dual}+1) > \epsilon)$ **or** $(\Delta\boldsymbol{\lambda}(t_{dual}+1) > \epsilon)$ **then**
11: $t_{dual} \leftarrow t_{dual}+1$
12: **go to** 2
13: **end if**
14: **return** $\boldsymbol{\nu}(t_{dual}+1), \boldsymbol{\lambda}(t_{dual}+1)$
15: **end function**

Theorem 5.1: For each cell i, the optimal solution to the resource allocation problem (12) is given by: $\theta_{k,n} = \frac{1}{\max(|\mathcal{K}(i)|, |\mathcal{N}|)}, \forall k \in \mathcal{K}(i), \forall n \in \mathcal{N}$.

The proof of Theorem 5.1 is given in Appendix I. When the number of active UEs is less than the number of available resources, $\theta_{k,n} = \frac{1}{|\mathcal{N}|}, \forall k \in \mathcal{K}(i), \forall n \in \mathcal{N}$. Thus, the available resources are not fully used over time, and each UE is permanently served. Otherwise, when $|\mathcal{K}(i)| > |\mathcal{N}|$, the optimal solution is: $\theta_{k,n} = \frac{1}{|\mathcal{K}(i)|}, \forall k \in \mathcal{K}(i), \forall n \in \mathcal{N}$. In this case, each RB is fully used over time, while UEs are not permanently served over time.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the convergence and the performance of our proposed centralized joint resource and power allocation problem.

A. Centralized Resource and Power Allocation

To verify the convergence of the centralized solution, we consider a multi-user OFDMA network, such as LTE/LTE-A networks, that consists of seven adjacent hexagonal cells, with one UE served by each cell. UE positions and radio conditions are randomly generated, and the initial power allocation for each RB equals π_{min}. System bandwidth equals 5 MHz. Thus, 25 RBs are available in each cell. The maximum transmission power per cell P_{max} is set to 43 dBm or 20 W. At the first iteration, the dual variables $\lambda_{k,i,n}(0), \forall k \in \mathcal{K}(i), \forall i \in \mathcal{I}, \forall n \in \mathcal{N}$, and $\nu_i(0), \forall i \in \mathcal{I}$, are assigned initial positive values. The evolution of $\widehat{\pi}_{i,1}$ along with the number of iterations is shown in Fig. 1, where $\widehat{\pi}_{i,1}$ is the logarithm of the transmission power allocated by the cell i to the RB 1. In addition, the number of primal iterations and the number of dual iterations per round are shown in Fig. 2.

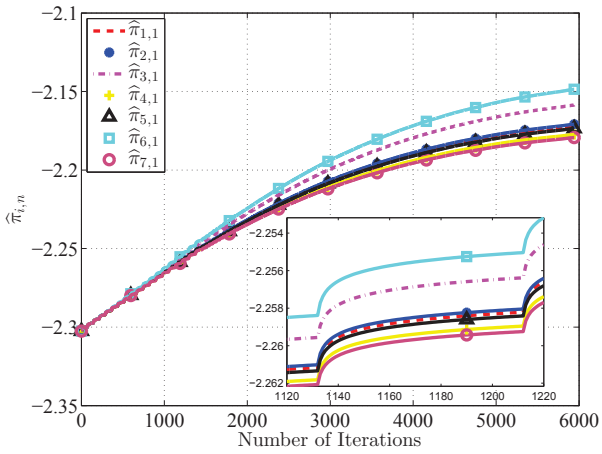


Fig. 1: Primal variables $\widehat{\pi}_{i,n}$

We notice that for the centralized power allocation approach, the primal problem requires approximately 6000 iterations to converge. As shown in Fig. 2, 1100 rounds are required to reach the optimal values of the primal and the dual variables. The zoomed box within Fig. 1 shows $\widehat{\pi}_{i,n}$ versus the number of primal iterations for a given round t. The values of $\widehat{\pi}_{i,n}$ are calculated using the dual variables obtained at the round $(t-$

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

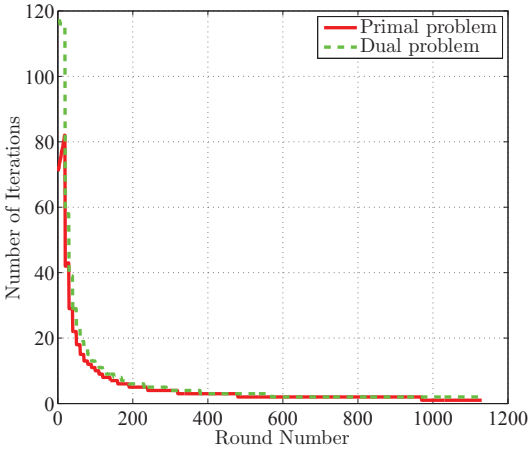


Fig. 2: Primal and dual iterations per round

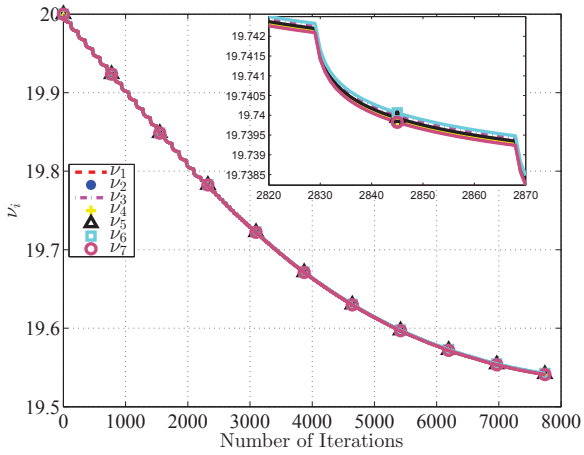


Fig. 4: Lagrange prices ν_i

1). We also notice that the number of primal iterations and the number of dual iterations decreases with the number of rounds. When t increases, the impact of Lagrange prices $\lambda_{k,i,n}(t)$ and $\nu_i(t)$ on the primal variables calculation is reduced, and the number of primal iterations required for convergence becomes lower. The same behavior is noticed for the number of dual iterations when the number of rounds increases.

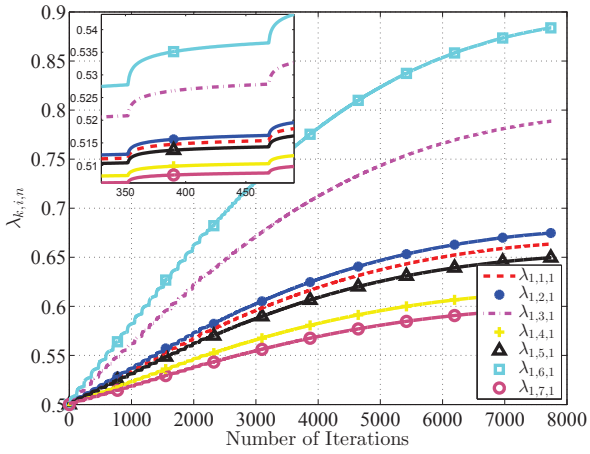


Fig. 3: Lagrange prices $\lambda_{k,i,n}$

For the same simulated scenario, we also show the dual variables $\lambda_{k,i,n}$ and ν_i versus the number of dual iterations in Fig. 3 and Fig. 4, respectively. We notice that approximately 8000 iterations are required for the dual problem to converge. At a given round t, the Lagrange prices $\lambda_{k,i,n}$ and ν_i are updated using the most recent values of the primal variables. The zoomed boxes within Fig. 3 and Fig. 4 show the evolution of $\lambda_{k,i,n}$ and ν_i versus the number of iterations, respectively. These values are updated until $\Delta\lambda_{k,i,n}$ and $\Delta\nu_i$ become less than ϵ. Convergence of the centralized power allocation problem occurs when two conditions are satisfied: first, the difference between the updated primal variables at round t and their values at round $(t-1)$ is less than ϵ. Second, the difference between the updated primal variables at round t and their values at round $(t-1)$ is less than ϵ.

B. Comparison with State-of-the-Art Approaches

We also compare the performance of our proposed centralized resource and power allocation approach with that of state-of-the-art resource and power allocation approaches such as the frequency reuse-1 model, the frequency reuse-3 model, FFR, and SFR techniques [26]. The frequency reuse-1 model allows the usage of the same frequency spectrum simultaneously in all the network cells. Moreover, homogeneous power allocation is performed.

In the frequency reuse-3 model, one third of the available spectrum is used in each cell in a cluster of three adjacent cells. Interference problems are eliminated, but the spectral efficiency is reduced. FFR and SFR techniques divide each cell into a cell-center and a cell-edge zones, and set restrictions on resource usage and power allocation in each zone. For all the compared techniques, resource allocation is performed according to Theorem 5.1.

1) System Throughput: For several simulation runs, we show the total system throughput for our proposed centralized resource and power allocation approach, for the frequency reuse-1 model, reuse-3 model, FFR, and SFR techniques under the same simulation scenarios. Simulation results, including the 95% confidence interval, are illustrated in Fig. 5.

It is shown that the centralized resource allocation approach offers the highest system throughput among all the compared techniques. In fact, it searches for the optimal resource and power allocation while taking into account restrictions on resource usage between the active UEs and on the downlink transmission power allocation. The achievable throughput is greater than that of the frequency reuse-3 model, FFR, and SFR techniques. Although the restrictions made on resource usage by these techniques mitigate ICI, the achievable throughput is reduced since the available spectrum in each cell or in each cell zone, is reduced.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

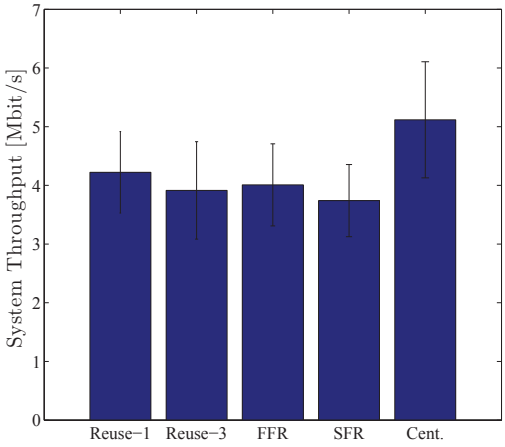


Fig. 5: System throughput

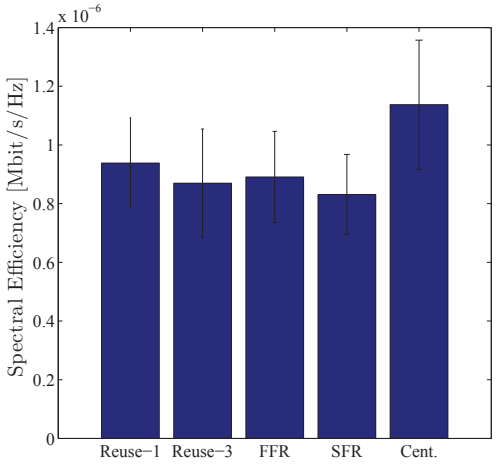


Fig. 6: Spectral efficiency

2) Spectral Efficiency: We also investigate the impact of the compared techniques on the spectral efficiency. Simulation results are shown in Fig. 6. Our proposed centralized resource allocation approach offers the highest spectral efficiency, since the optimal resource and power allocation is guaranteed. In fact, the static resource allocation between cell zones, and the quantified transmission power levels do not allow to perform flexible resource allocation in a manner that satisfies UE needs in each cell.

VII. Conclusion

Resource and power allocation problem is a challenging problem for present and future wireless networks. Several state-of-the-art techniques consider the joint resource and power allocation problem, and formulate it as nonlinear optimization problems. However, the main disadvantage of these techniques is that they do not consider the impact of inter-cell interference. Indeed, each cell solves its own resource and power allocation problem without taking into account resource usage and power allocation in the neighboring cells.

In this paper, we formulated the multi-cell joint resource and power allocation problem for multiuser OFDMA networks as a centralized optimization problem, where the objective is to maximize system throughput while guaranteeing throughput fairness between UEs. The joint problem is then decomposed into two independent problems: a resource allocation problem and a power allocation problem. Contrarily to the majority of the state-of-the-art approaches, inter-cell interference is not neglected, and the impact of the simultaneous transmissions in the neighboring cells is taken into account when managing the resource and power allocation. Simulation results prove the convergence of the optimization variables, and show the positive impact of our proposed centralized resource and power allocation approach on system performance.

Appendix I

Proof of Theorem 5.1

The objective function (12a), can be written as follows:

$$(\eta_2)_i \quad = \log\left(\prod_{\substack{k\in\mathcal{K}(i)\\ n\in\mathcal{N}}} \theta_{k,n}\right). \tag{19}$$

Since the logarithmic function is monotonically increasing, the maximization of $(\eta_2)_i$ is equivalent to the maximization of the term $\prod_{\substack{k\in\mathcal{K}(i)\\ n\in\mathcal{N}}} \theta_{k,n}$. We consider the following cases:

1) Let us assume that:

$$\sum_{k\in\mathcal{K}(i)} \theta_{k,n} < \sum_{n\in\mathcal{N}} \theta_{k,n},\ \forall\ k\ \in \mathcal{K}(i),\ \forall\ n \in\ \mathcal{N}. \tag{20}$$

We suppose that $\theta_{k,n}, \forall\ k \in \mathcal{K}(i), \forall\ n \in \mathcal{N}$ is an optimal solution to the resource allocation problem (12) $i.e.$, this solution maximizes the objective function (12a). For this solution, we assume that:

$$\exists\ k\ \in\ \mathcal{K}(i)\ /\ \sum_{n\in\mathcal{N}} \theta_{k,n} < 1. \tag{21}$$

We define $\epsilon > 0$ as follows:

$$\epsilon = 1 - \sum_{n\in\mathcal{N}} \theta_{k,n},$$

and we demonstrate that this solution is not an optimal solution to problem (12) using the proof by contradiction. In fact, we define a set of $\theta'_{k,n}$ variables as follows:

$$\theta'_{k,n} = \begin{cases} \theta_{k,n}, & \forall\ n \in \mathcal{N},\ n \neq n_1, \forall\ k \in \mathcal{K}(i), \\ \theta_{k,n} + \epsilon, & \text{if } n = n_1, \forall\ k \in \mathcal{K}(i). \end{cases}$$

Therefore, we have:

$$\prod_{\substack{k\in\mathcal{K}(i)\\ n\in\mathcal{N}}} \theta'_{k,n} = \prod_{\substack{k\in\mathcal{K}(i)\\ n\in\mathcal{N}}} \theta_{k,n} + \epsilon \cdot \prod_{\substack{k\in\mathcal{K}(i)\\ n\in\mathcal{N}}} \theta_{k,n} > \prod_{\substack{k\in\mathcal{K}(i)\\ n\in\mathcal{N}}} \theta_{k,n},$$

and the assumption made in (21) is false, since it does not maximize the objective function (12a). Con-

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

sequently, we have:

$$\sum_{n\in\mathcal{N}} \theta_{k,n} = 1, \forall\ k\ \in\ \mathcal{K}(i),$$
$$\Rightarrow \quad \sum_{k\in\mathcal{K}(i)} \sum_{n\in\mathcal{N}} \theta_{k,n} = |\mathcal{K}(i)|.$$

Since the sum of all the $\theta_{k,n}$ variables is constant, the term $\prod_{\substack{k\in\mathcal{K}(i)\\ n\in\mathcal{N}}} \theta_{k,n}$ reaches its maximum when all the variables $\theta_{k,n}$ are equal $i.e.$,

$$\theta_{k,n} = \frac{|\mathcal{K}(i)|}{|\mathcal{K}(i)|\cdot|\mathcal{N}|} = \frac{1}{|\mathcal{N}|}, \forall\ k \in \mathcal{K}(i), \forall\ n \in \mathcal{N},$$

which is an optimal solution to the resource allocation problem (12).

2) Similarly, when:

$$\sum_{n\in\mathcal{N}} \theta_{k,n} < \sum_{k\in\mathcal{K}(i)} \theta_{k,n},\ \forall\ k\ \in \mathcal{K}(i),\ \forall\ n \in\ \mathcal{N}. \quad (22)$$

In this case, the optimal solution is given by:

$$\theta_{k,n} = \frac{|\mathcal{N}|}{|\mathcal{K}(i)|\cdot|\mathcal{N}|} = \frac{1}{|\mathcal{K}(i)|}, \forall\ k \in \mathcal{K}(i), \forall\ n \in \mathcal{N}.$$

REFERENCES

[1] 3GPP, "Physical Layer Aspects for Evolved Universal Terrestrial Radio Access (UTRA) (Release 7)," 3GPP TR 25.814 V7.1.0, Tech. Rep., 2006.

[2] ——, "Evolved Universal Terrestrial Radio Access (E-UTRA): Physical Layer Procedures," 3GPP TS 36.213 V11.11.0, Technical Specification, December 2013.

[3] N. Hassan and M. Assaad, "Optimal Fractional Frequency Reuse (FFR) and Resource Allocation in Multiuser OFDMA System," in *Int. Conf. Information and Communication Technologies*, Karachi, August 2009, pp. 88–92.

[4] Huawei, "Soft Frequency Reuse Scheme for UTRANLTE (R1-050507)," 3GPP RAN WG1 no. 41, Athens, Greece, Technical report, May 2005.

[5] M. Yassin, S. Lahoud, M. Ibrahim, and K. Khawam, "A Downlink Power Control Heuristic Algorithm for LTE Networks," in 21^{st} *Int. Conf. Telecommunications*, Lisbon, May 2014, pp. 323–327.

[6] M. Yassin, S. Lahoud, M. Ibrahim, K. Khawam, D. Mezher, and B. Cousin, "Non-Cooperative Inter-Cell Interference Coordination Technique for Increasing Through Fairness in LTE Networks," in *IEEE* 81^{st} *Vehicular Technology Conf.*, Glasgow, May 2015.

[7] M. Chiang, C. W. Tan, D. Palomar, D. O'Neill, and D. Julian, "Power Control By Geometric Programming," *IEEE Trans. Wireless Commun.*, vol. 6, no. 7, pp. 2640–2651, July 2007.

[8] J. Tang, D. So, E. Alsusa, and K. Hamdi, "Resource Efficiency: A New Paradigm on Energy Efficiency and Spectral Efficiency Tradeoff," *IEEE Trans. Wireless Commun.*, vol. 13, no. 8, pp. 4656–4669, Aug 2014.

[9] R. Loodaricheh, S. Mallick, and V. Bhargava, "Energy-Efficient Resource Allocation for OFDMA Cellular Networks with User Cooperation and QoS Provisioning," *IEEE Trans. Wireless Commun.*, vol. 13, no. 11, pp. 6132–6146, Nov. 2014.

[10] C. Xiong, G. Li, S. Zhang, Y. Chen, and S. Xu, "Energy and Spectral-Efficiency Tradeoff in Downlink OFDMA Networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 11, pp. 3874–3886, November 2011.

[11] T. Quek, Z. Lei, and S. Sun, "Adaptive Interference Coordination in Multi-Cell OFDMA Systems," in *IEEE* 20^{th} *Int. Symp. Personal, Indoor and Mobile Radio Commun.*, September 2009, pp. 2380–2384.

[12] Y. Umeda and K. Higuchi, "Efficient Adaptive Frequency Partitioning in OFDMA Downlink with Fractional Frequency Reuse," in *Int. Symp. Intelligent Signal Processing and Commun. Systems*, December 2011, pp. 1–5.

[13] L. Venturino, A. Zappone, C. Risi, and S. Buzzi, "Energy-Efficient Scheduling and Power Allocation in Downlink OFDMA Networks with Base Station Coordination," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 1–14, January 2015.

[14] C. Xiong, G. Li, S. Zhang, Y. Chen, and S. Xu, "Energy-Efficient Resource Allocation in OFDMA Networks," *IEEE Trans. Commun.*, vol. 60, no. 12, pp. 3767–3778, December 2012.

[15] J. Yu, G. Li, C. Yin, S. Tang, and X. Zhu, "Multi-Cell Coordinated Scheduling and Power Allocation in Downlink LTE-A Systems," in *IEEE* 80^{th} *Vehicular Technology Conf.*, Vancouver, September 2014, pp. 1–5.

[16] M. Aboul Hassan, E. Sourour, and S. Shaaban, "Novel Resource Allocation Algorithm for Improving Reuse One Scheme Performance in LTE Networks," in 21^{st} *Int. Conf. Telecommunications*, Lisbon, May 2014, pp. 166–170.

[17] D. T. Ngo, S. Khakurel, and T. LeNgoc, "Joint Subchannel Assignment and Power Allocation for OFDMA Femtocell Networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 1, pp. 342–355, Jan. 2014.

[18] J. Zheng, Y. Cai, Y. Liu, Y. Xu, B. Duan, and X. Shen, "Optimal Power Allocation and User Scheduling in Multicell Networks: Base Station Cooperation Using a Game-Theoretic Approach," *IEEE Trans. Wireless Commun.*, vol. 13, no. 12, pp. 6928–6942, Dec. 2014.

[19] S. Sadr and R. Adve, "Partially-Distributed Resource Allocation in Small-Cell Networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 12, pp. 6851–6862, Dec. 2014.

[20] J. S. Wang, J. H. Lee, J. C. Park, I. Song, and Y. H. Kim, "Combining of Cyclically Delayed Signals: A Low-Complexity Scheme for PAPR Reduction in OFDM Systems," *IEEE Trans. Broadcasting*, vol. 56, no. 4, pp. 577–583, December 2010.

[21] E. Dahlman, S. Parkvall, and J. Skold, *4G LTE and LTE-Advanced for Mobile Broadband*, 1st ed., Elsevier, Ed. Oxford: Elsevier, 2011.

[22] F. Kelly, "Charging and Rate Control for Elastic Traffic," *European Trans. Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*, 7th ed., Cambridge University Press, Ed. Cambridge: UK: Cambridge University Press, 2009, vol. 7.

[24] D. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE J. Selected Areas in Commun.*, vol. 24, no. 8, pp. 1439–1451, August 2006.

[25] M. Chiang, "Geometric Programming for Communication Systems," *Foundations and Trends of Commun. and Information Theory*, vol. 2, no. 1-2, pp. 1 – 156, August 2005.

[26] M. Yassin, M. AboulHassan, S. Lahoud, M. Ibrahim, D. Mezher, B. Cousin, and E. Sourour, "Survey of ICIC Techniques in LTE Networks under Various Mobile Environment Parameters," *Accepted for publication in Springer Wireless Networks*, 2015.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Content-Centric Networking Using Anonymous Datagrams

J.J. Garcia-Luna-Aceves[1,2] and Maziar Mirzazad Barijough[2]
[1]Palo Alto Research Center, Palo Alto, CA 94304
[2]Department of Computer Engineering, University of California, Santa Cruz, CA 95064
Email: jj@soe.ucsc.edu, maziar@soe.ucsc.edu

***Abstract*—Using Interests (requests that elicit content) and maintaining per-Interest forwarding state in Pending Interest Tables (PIT) are integral to the design of the Named Data Networking (NDN) and Content-Centric Networking (CCNx) architectures. However, using PITs makes the network vulnerable to Interest-flooding attacks, and PITs can become very large. It is shown that in-network caching eliminates the need for Interest aggregation and obviates the use of PITs. A new approach to content-centric networking (CCN-GRAM) is introduced that provides all the benefits of NDN and CCNx, eliminates the use of PITs by means of anonymous datagrams, and is immune to Interest-flooding attacks. Routers maintain routes to the anonymous origins of Interests using an on-demand routing approach in the data plane that can also be used to provide native support for multicasting in the dat a plane. Simulation experiments show that the number of forwarding entries required in CCN-GRAM is orders of magnitude smaller than the number of PIT entries.**

I. Introduction

The leading approach in content-centric networking consists of: populating forwarding information bases (FIB) maintained by routers with routes to name prefixes denoting content, sending content requests (called Interests) for specific content objects (CO) over paths implied by the FIBs, and delivering data packets with content objects along the reverse paths traversed by Interests.

The main advantages that such Interest-based content-centric networking approach offers compared to the IP Internet are that: (a) content providers and caching sites do not know the identity of the consumers requesting content; (b) content can be obtained by name from those sites that are closer to consumers; (c) data packets carrying content cannot traverse loops, because they are sent over the reverse paths traversed by Interests; and (d) content-oriented security mechanisms can be implemented as part of the content delivery mechanisms.

Named data networking (NDN) [18] and CCNx [5] are the two prominent Interest-based content-centric networking approaches. Routers in NDN and CCNx maintain a "stateful forwarding plane" [27] (i.e., per-Interest forwarding state) by means of Pending Interest Tables (PIT). The PIT of a router maintains information regarding the incoming interfaces from which Interests for a CO were received and the interfaces where the Interest for the same CO was forwarded.

ISBN 978-3-901882-83-8

Since the inception of CCNx and NDN, PITs have been viewed as necessary in order to maintain routes to the origins of Interests while preserving the anonymity of those sources, aggregate Interests requesting the same content in order to attain efficient Interest and content forwarding, and support multicasting without additional support in the control plane.

However, using PITs at Internet scale comes at a big price. PITs grow very large [8], [21], [22] as the number of Interests from users increases, which results from PITs having to store per-Interest forwarding state. Furthermore, PITs make routers vulnerable to Interest-flooding attacks [16], [23], [25], [26] in which adversaries send malicious Interests aimed at making the size of PITs explode. Known countermeasures to these attacks [2] attempt to reduce the rates at which suspected routers can forward Interests. However, these solutions cannot prevent all flooding attacks and can actually be used to mount other types of denial-of-service attacks.

Section II analyzes the effectiveness of Interest aggregation in NDN by means of simulations based on the implementation of NDN in ndnSIM [1] without modifications. The results show that the percentage of Interests that are aggregated is negligible when in-network caching is enabled, even when Interests exhibit temporal or spatial correlation.

Given that in-network caching obviates the need for Interest aggregation, and given the vulnerability of NDN and CCNx to Interest-flooding attacks, it is clear that a new Interest forwarding approach is needed for content-centric networking.

We present **CCN-GRAM** (*Gathering of Routes for Anonymous Messengers*), which provides all the benefits of content-centric networking, including native support for multicasting in the data plane, and eliminates the need to maintain per-Interest forwarding state by forwarding Interests and responses to them using *anonymous* datagrams.

Section III describes the operation of CCN-GRAM. Like NDN and CCNx, CCN-GRAM uses Interests, data packets, and replies to Interests. Similar to IP datagrams, the messages sent in CCN-GRAM specify a source and a destination. For an Interest, the source of an Interest is an anonymous identifier with local context and the destination is the name of a content object. For data packets and replies to Interests, the source is the name of a content object and the destination is an anonymous identifier. A novel on-demand routing approach is used to maintain routes to the anonymous routers that

originate Interests for specific content on behalf of local content consumers. Only the local router serving a user knows the identity of the user; no other router, content provider, or caching site can determine the consumer that originated an Interest, without routers collaborating along the path traversed by the Interests to establish the provenance of the Interest.

In contrast to NDN and CCNx in which Interests may traverse forwarding loops [11], [12], [14], forwarding loops cannot occur in CCN-GRAM for either Interests or responses sent to Interests, even if the FIBs maintained by routers contain inconsistent forwarding state involving routing-table loops. Furthermore, the anonymous datagram forwarding of CCN-GRAM is much simpler than the label-swapping approach we have advocated before [13], [15].

Forwarding of Interests and responses to them in CCN-GRAM uses four tables: a LIGHT (Local Interests GatHered Table), a FIB, an ART (Anonymous Routing Table) and a LIST (Local Interval Set Table). The LIGHT of a router is an index listing content that is locally available and content that is remote and has been requested by *local* users. The FIB of each router states the next hops to each name prefix and the distance to the name prefix reported by each next hop. The ART is maintained using Interests and states the paths to destinations denoted with local identifiers from which routers cannot discern the origin of Interests. The LIST states the intervals of local identifiers that a router assigns to its neighbors and that each neighbor assigns to the router.

Section IV compares the performance of CCN-GRAM with NDN when routes to name prefixes are static and loop-free, which is the best case for NDN. The network consists of 150 routers, with 10 being connected to content producers and 50 being connected to consumers. CCN-GRAM attains similar end-to-end latencies than NDN in retrieving content. However, depending on the rate at which Interests are submitted, CCN-GRAM requires an average number of forwarding entries per router that is 5 to more than150 times smaller than the number of PIT entries needed in NDN.

II. Interest Aggregation in NDN

A. Elements of NDN Operation

Routers in NDN use Interests, data packets, and negative acknowledgments (NACK) to exchange content. An Interest is identified in NDN by the name of the CO requested and a nonce created by the origin of the Interest. A data packet includes the CO name, a security payload, and the payload itself. A NACK carries the information needed to denote an Interest and a code stating the reason for the response.

A router r uses three data structures to process Interests, data packets, and NACKs: A content store (CS), a forwarding information base (FIB), and a pending Interest table (PIT). A CS is a cache for COs indexed by their names. With on-path caching, routers cache the content they receive in response to Interests they forward.

A FIB is populated using content routing protocols [10], [17] or static routes and a router matches Interest names stating a specific CO to FIB entries corresponding to prefix names using *longest prefix match*. The FIB entry for a given name prefix lists the interfaces that can be used to reach the prefix. In NDN, a FIB entry also contains additional information [18].

The entry in a PIT for a given CO consists of one or multiple tuples stating a nonce received in an Interest for the CO, the incoming interface where it was received, and a list of the outgoing interfaces over which the Interest was forwarded.

When a router receives an Interest, it checks whether there is a match in its CS for the CO requested in the Interest. The Interest matching mechanisms used can vary, and for simplicity we focus on exact Interest matching only. If a match to the Interest is found, the router sends back a data packet over the reverse path traversed by the Interest. If no match is found in the CS, the router determines whether the PIT stores an entry for the same content.

In NDN, if the Interest states a nonce that differs from those stored in the PIT entry for the requested content, then the router "aggregates" the Interest by adding the incoming interface from which the Interest was received and the nonce to the PIT entry without forwarding the Interest. If the same nonce in the Interest is already listed in the PIT entry for the requested CO, the router sends a NACK over the reverse path traversed by the Interest. If a router does not find a match in its CS and PIT, the router forwards the Interest along a route (or routes) listed in its FIB for the best prefix match. In NDN, a router can select an interface to forward an Interest if it is known that it can bring content and its performance is ranked higher than other interfaces that can also bring content.

B. Likelihood of Interest Aggregation in NDN

We analyze the likelihood that interest aggregation occurs in the presence of in-network caching in NDN using simulations carried out with the NDN implementation in ndnSIM [1]without modifications. A more detailed analysis is presented in [7]. Our study is independent of the Interest retransmission strategy, and uses the percentage of aggregated Interests in the network as the performance metric. For simplicity, we assume that routers use exact Interest matching to decide whether an Interest can be answered.

1) Scenario Parameters: We consider the average latencies between routers, the capacity of caches, the Interest request rates from routers, the popularity of content, and the temporal correlation of content requests. The scenarios we use consist of random networks with 200 nodes corresponding to routers distributed uniformly in a 100m×100m area. Routers with 12m or shorter distance are connected to each other with a point-to-point link, which results in a topology with 1786 edges. Each router acts as a producer of content and also has local consumers generating Interests.

Producers are assumed to publish 1,000,000 different content objects that are uniformly distributed among routers. For simplicity, we assume that *all* routers have the same storage capacity in their caches, which depending on the experiment ranges from 0 to up to 100,000 cache entries per router, or 10% of the published objects.

The distribution of object requests determines how many Interests from different users request the same content. It has been argued [9] that Internet traffic follows a Zipf distribution with a parameter (α) value close to 1. A smaller Zipf parameter value results in a lower Interest aggregation amount. Accordingly, we model object popularity using a Zipf distribution with values of α equal to 0.7 and 1.

We considered different values of the *total Interest rate per router*, corresponding to the sum of Interests from all local users. Increasing values of Interest rates can be viewed as higher request rates from a constant user population of local active users per router, or an increasing population of active users per router. For example, 50 to 500 Interests per second per router can be just 10 Interests per second per active user for a local population of 5 to 50 concurrently active users per router. The Interest rates we assume per router are not large compared to recent results on the size that PITs would have in realistic settings [8], [22], [23], [21].

The percentage of Interests that benefit from Interest aggregation is a function of the RTT between the router originating the Interest and the site with the requested content, as well as the PIT entry expiration time when the Interest is not answered with a data packet or a NACK. Recent Internet latency statistics [3] show that Internet traffic latency varies from 11ms for regional European traffic to 160ms for long-distance traffic. Accordingly, we consider point-to-point delays of 10ms between neighbor routers in many of our simulations, which leads to RTTs of about 200ms. We also carried experiments varying the RTT of the network below and above 200ms.

2) Simulation Results: The following simulation results can be viewed as applicable to the steady-state behavior of a network using NDN.

Figure 1 shows the effect of the α parameter and RTTs when the request rate per router is 50 Interests per second. The latencies between neighbor routers are set to 5 and 15 ms, which produce RTTs of 66 to 70 ms and 193 to 200 ms, respectively. It is clear that Interest aggregation is far less important when consumers are less likely to request similar content ($\alpha = 0.7$). Furthermore, the benefits of Interest aggregation vanish as caches are allowed to cache more content. When caches can store up to 1% of the total number of objects published, the percentage of Interests that are aggregated is less than 2% for $\alpha = 1$ and less than 0.8% for $\alpha = 0.7$.

In theory, Interest aggregation is most useful when Interests exhibit temporal correlation, such as when popular live events take place. Figure 2 shows the impact of caching on Interest aggregation when Interests have temporal correlation and either no caching is used or caches with capacity for only 0.1% of the objects published in the network are used. Interests are generated using the model proposed by Dabirmoghaddam et al [6] with a Zipf parameter value of $\alpha = 0.7$ and results for three total Interest rates per router and four temporal localization factors for Interests are shown. A higher temporal locality factor indicates a higher degree of popularity of objects in the same time period. The results in Figure 2 show that, without caching, Interest aggregation *is* very important for all values of temporal locality of Interest popularity, and is more important when Interest locality is high (large localization factor). However, once caching is allowed and even if caches can store only up to 0.1% of the published objects, the percentage of aggregated Interests is minuscule and actually decreases with the temporal correlation of Interests.

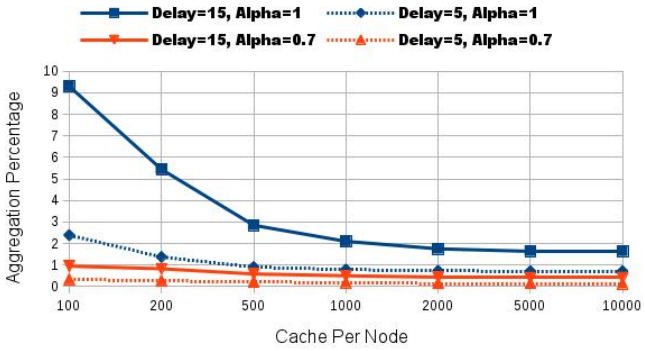


Fig. 1. Interest aggregation as a function of values of Zipf parameter, caching capacity, and RTTs

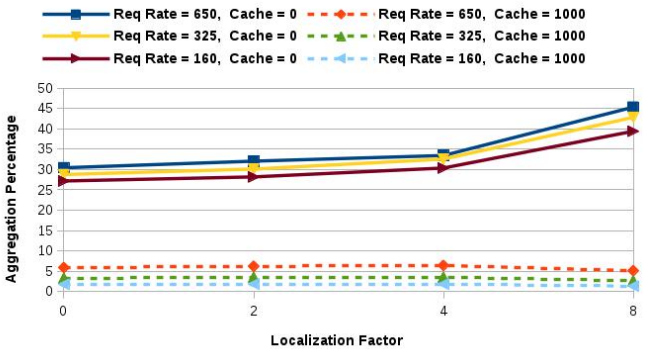


Fig. 2. Percentage of Interest aggregation under temporal locality

III. CCN-GRAM

We assume that Interests are retransmitted only by the consumers that originated them. We assume that routers use exact Interest matching, and that a router that advertises being an origin of a name prefix stores all the content objects associated with that prefix at a local content store. Routers know which interfaces are neighbor routers and which are local users, and forward Interests on a best-effort basis. For convenience, it is assumed that a request for content from a local user is sent to its local router in the form of an Interest.

A. Information Exchanged and Stored

The name of content object (CO) j is denoted by $n(j)$ and the name prefix that is the best match for name $n(j)$ is denoted by $n(j)^*$. The set of neighbors of router i is denoted by N^i.

An Interest forwarded by router k requesting CO $n(j)$ is denoted by $I[n(j), AID^I(k), D^I(k)]$, and states the name of the requested CO ($n(j)$), an anonymous identifier ($AID^I(k)$) used to denote the origin of the Interest, and the distance from k to the requested content.

A data packet sent by router i in response to an Interest is denoted by $DP[n(j), AID^R(i), sp(j)]$, and states the name of

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

the CO being sent ($n(j)$), an anonymous identifier ($AID^R(i)$) that states the intended recipient of the data packet, and a security payload ($sp(j)$) used optionally to validate the CO.

A reply sent by router i in response to an Interest is denoted by $REP[n(j),\ AID^R(i),\ \mathsf{CODE}]$ and states the name of a CO ($n(j)$), an anonymous identifier ($AID^R(i)$) that states the intended recipient of the reply, and a code (CODE) indicating the reason why the reply is sent. Possible reasons for sending a reply include: an Interest loop is detected, no route is found towards requested content, and no content is found.

Router i maintains four tables for forwarding: an optional Local Interests Gathered Table ($LIGHT^i$), a forwarding information base (FIB^i), an anonymous routing table (ART^i), and a Local Interval Set Table ($LIST^i$).

$LIGHT^i$ lists the names of the COs requested by router i or already stored at router i. It is indexed by the CO names that have been requested by the router on behalf of local customers. The entry for CO name $n(j)$ states the name of the CO ($n(j)$), a pointer to the content of the CO ($p[n(j)]$), and a list of zero or more identifiers of local consumers ($lc[n(j)]$) that have requested the CO while the content is remote.

FIB^i is indexed using known content name prefixes. The entry for prefix $n(j)^*$ states the distance reported by each next-hop neighbor router for the prefix. The distance stored for neighbor q for prefix $n(j)^*$ in FIB^i is denoted by $D(i, n(j)^*, q)$. Each entry in FIB^i is stored for a maximum time determined by the lifetime of the corresponding entry in the routing table of the router.

$LIST^i$ maintains the intervals of anonymous identifiers used by router i. It states the local interval of identifiers accepted by router i (denoted by $LI^i(i)$), and the local interval of identifiers accepted by each neighbor router k (denoted by $LI^i(k)$) . Clearly, $LI^i(k) = LI^k(k)$. All local intervals have the same length $|LI|$.

ART^i is indexed using the anonymous identifiers taken from $LI^i(i)$. Each entry states an anonymous identifier of a destination ($AID(ART^i)$), a next hop to the destination, ($s(ART^i)$), and an identifier mapping used to handle identifier collisions ($map(ART^i)$). $ART^i[AID, s, map]$ is used to denote a given entry in ART^i.

Routers can exchange local intervals with their neighbors in a number of ways. The exchange can be done in the data plane using Interests and data packets. An example would be having a router send an Interest stating a common name denoting that a local interval is requested, and an empty AID. Given the succinct way in which local intervals can be stated (an identifier denotes its interval), the exchange can also be easily done as part of the routing protocol running in the control plane. Routers could exchange interval identifiers in HELLO messages, link-state advertisements or distance updates. To simplify our description of CCN-GRAM, we assume that routers have exchanged their local intervals with one another and have populated their LISTs accordingly. We also assume that local intervals do not change for extended periods of time after they are assigned.

B. Eliminating Forwarding Loops

Let $S^i_{n(j)^*}$ denote the set of next-hop neighbors of router i for prefix $n(j)^*$. The following rule is used to ensure that Interests cannot traverse routing loops, even if the routing data stored in FIBs regarding name prefixes is inconsistent and leads to routing-table loops.

Loop-Free Forwarding Rule (LFR):
Router i accepts $I[n(j), AID^I(k), D^I(k)]$ from router k if:

$$\exists\ v \in S^i_{n(j)^*}(\ D^I(k) > D(i, n(j)^*, v)\ ) \qquad (1)$$

LFR is based on the same invariants we have proposed previously to eliminate Interest looping in NDN and CCNx [11], [14] and avoid forwarding loops in more efficient forwarding planes for content-centric networks [13]. As we explain in [11], [13], [14], the approach is a simple application of diffusing computations that ensures loop-free forwarding of Interests with or without aggregation.

C. Forwarding to Anonymous Destinations

The header of a datagram needs to denote its origin and destination, so that the datagram can be forwarded to the intended destination and responses to the datagram can be forwarded back to the source. Since the introduction of datagram packet switching by Baran [4], the identifiers used to denote the sources and destinations of datagrams have had global scope, and routers maintain FIBs with entries towards those sources. However, this need not be the case!

It is trivial to add information in Interests about the paths they traverse (e.g., see [20]) to allow responses to be sent back without the need for FIBs maintaining routes to the sources of Interests. However, this would negate the anonymity of Interests advocated in NDN and CCNx.

CCN-GRAM uses local identifiers to denote the sources of Interests in a way that responses to Interests (data packets or replies) can be forwarded correctly to the sources of Interests, without their identity being revealed to relaying routers, caching sites, or content producers.

Given that all local intervals have the same length $|LI|$, the local interval $LI^i(i)$ is uniquely defined by the smallest identifier of the interval, which we denote by $LI^i(i)[s]$.

If router p sends Interest $I[n(j), AID^I(p), D^I(p)]$ to router i, $AID^I(p)$ must be in $LI^p(i) = LI^i(i)$. Similarly, if router i forwards Interest $I[n(j), AID^I(i), D^I(i)]$ to router n, $AID^I(i)$ must be in $LI^i(n) = LI^n(n)$. Hence, to forward an Interest from p to n, router i must map the AID received in the Interest from p to an AID that belongs to the local interval accepted by its neighbor n. Router i can accomplish this mapping with the following bijection, where ϵ is a constant known only to router i:

$$AID^I(i) = \epsilon + AID^I(p) - LI^i(i)[s] + LI^i(n)[s]\ mod\ |LI| \quad (2)$$

We denote the mapping of identifiers from $LI^i(i)$ to $LI^i(n)$ by $f_i(n) : LI^i(i) \rightarrow LI^i(n)$. The image of identifier $a \in LI^i(i)$ under $f_i(n)$ is denoted by $f_i(n)[a]$ and $f_i(n)[a] \in LI^i(n)$. The reverse mapping from $LI^i(n)$ to $LI^i(i)$ is denoted by $f_i^{-1}(n)$ and of course $f_i^{-1}(n)[f_i(n)[a]] = a$.

Algorithms 1 to 4 specify the steps taken by routers to process and forward Interests, and return data packets or replies. We assume that each router is initialized properly, knows the identifiers used to denote local consumers, knows all its neighbors, and knows the local identifier intervals associated with each neighbor. We assume that a routing protocol (e.g., DCR [10], NLSR [17]) operating in the control plane updates the entries of routing tables listing one or multiple next hops towards name prefixes. Routers populate their FIBs with routes to name prefixes based on the data stored in their routing tables. How long FIB entries are maintained is determined by the operation of the routing protocol.

We assume that router i uses a single anonymous identifier in $LI^i(i)$ to denote itself in its ART^i, and denote it by AID^i.

Algorithm 1 Processing Interest from user c at router i

function Interest_Source
INPUT: $LIGHT^i$, $LIST^i$, FIB^i, ART^i, AID^i, $I[n(j), c, nil]$;
if $n(j) \in LIGHT^i$ **then**
  if $p[n(j)] \neq nil$ (% CO is local) **then**
    retrieve CO $n(j)$; send $DP[n(j), c, sp(j)]$ to consumer c
  else
    $p[n(j)] = nil$; $lc[n(j)] = lc[n(j)] \cup c$ (% Interest is aggregated)
  end if
else
  if $n(j)^* \in LIGHT^i$ (%All content in $n(j)^*$ is local and $n(j)$ is not) **then**
    send $REP[n(j), c, \text{no content}]$ (% $n(j)$ does not exist)
  else
    if $n(j)^* \notin FIB^i$ **then**
      send $REP[n(j), \text{no route}, c]$ to c (% No route to $n(j)^*$ exists)
    else
      create entry for $n(j)$ in $LIGHT^i$: (% Interest from c is recorded)
      $lc[n(j)] = lc[n(j)] \cup c$; $p[n(j)] = nil$;
      if $AID^i = nil$ **then**
        select identifier $a \in LI^i(i)$ that is not used in any entry in ART^i;
        $AID^i = a$; create entry $ART^i[AID^i, i, AID^i]$
      end if
      for each $v \in N^i$ **by rank in** FIB^i **do**
        $AID^I(i) = f_i(v)[AID^i]$; $D^I(i) = D(i, n(j)^*, v)$;
        send $I[n(j), AID^I(i), D^I(i)]$ to v; **return**
      end for
    end if
  end if
end if

Algorithm 1 shows the steps taken by router i to process Interests received from local consumers. For convenience, content requests from local consumers are assumed to be Interests stating the name of a CO, the name of the consumer, and an empty distance to the content assumed to denote infinite. Similarly the same format of data packets and replies used among routers is used to denote the responses a router sends to local consumers.

After receiving an Interest from a local consumer, router i first searches its LIGHT to determine if the content is stored locally or a request for the same content is pending. If the content is stored locally, a data packet is sent back to the user requesting the CO. If a request for the same content is pending, the name of the user is simply added to the list of users that have requested the CO.

In our description of CCN-GRAM, a router that advertises being an origin of a prefix must have all the COs associated with the prefix stored locally. If router i states that it is an origin of the name prefix $n(j)^*$ and a specific CO with a name that is in that prefix is not found locally, a reply must be sent back to the consumer stating that the content does not exist. Additional steps could be taken to address the case of Interests sent maliciously for content that does not exist.

If the CO is remote and no FIB entry exists for a name prefix that can match $n(j)$, a reply is sent back stating that no route to the CO could be found. Otherwise, router i forwards the Interest through the highest ranked neighbor v in its FIB for the name prefix matching $n(j)$, which is denoted by $n(j)^*$. How such a ranking is done is left unspecified, and can be based on a distributed or local algorithm [10], [17], [19].

When router i originates an Interest on behalf of a local consumer and forwards Interest $I[n(j), AID^I(i), D^I(i)]$ to neighbor router n towards name prefix $n(j)^*$, router i selects an identifier $a \in LI^i(i)$ that is not used to denote any other source of Interests in ART^i, sets $AID^I(i) = f_i(n)[a] \in LI^i(n)$, and stores the entry $ART^i[a, i, a]$. Router i can use the same anonymous identifier for all the Interests it originates on behalf of local consumers and forwards to neighbor n.

If no ART entry exists with router i as the origin of Interests ($AID^i = nil$), AID^i is selected from the set of AIDs in $LI^i(i)$ that are not being used for other Interest sources, and a new ART entry is created for AID^i. The Interest is forwarded to the selected next hop for the Interest by first mapping AID^i into an AID in $LI^i(v)$ using the bijection in Eq. 2.

Algorithm 2 Processing Interest from router p at router i

function Interest_Forwarding
INPUT: $LIGHT^i$, $LIST^i$, FIB^i, ART^i, $I[n(j), AID^I(p), D^I(p)]$;
$AID^R(i) = AID^I(p)$;
if $n(j) \in LIGHT^i$ **then**
  if $p[n(j)] \neq nil$ **then**
    retrieve CO $n(j)$; send $DP[n(j), AID^R(i), sp(j)]$ to p
  end if
else
  if $n(j)^* \in LIGHT^i$ **then**
    send $REP[n(j), AID^R(i), \text{no content}]$ to p (% $n(j)$ does not exist)
  else
    if $n(j)^* \notin FIB^i$ **then**
      send $REP[n(j), AID^R(i), \text{no route}]$ to p (% No route to $n(j)^*$ exists)
    else
      for each $s \in N^i$ **by rank in** FIB^i **do**
        if $D^I(p) > D(i, n(j)^*, s)$ (% LFR is satisfied) **then**
          $SET = \emptyset$; $AID^I(i) = nil$; $collision = 0$;
          for each entry $ART^i[AID, s, map]$ **do**
            $SET = SET \cup \{AID\}$;
            if $AID(ART^i) = AID^I(p)$ **then**
              if $s(ART^i) = p$ **then**
                $AID^I(i) = f_i(s)[AID(ART^i)]$
              else
                $collision = 1$
              end if
            end if
            if $map(ART^i) = AID^I(p) \wedge s(ART^i) = p$ **then**
              $AID^I(i) = f_i(s)[AID(ART^i)]$
            end if
          end for
          if $collision = 0 \wedge AID^I(i) = nil$ **then**
            create entry $ART^i[AID^I(p), p, AID^I(p)]$;
            $AID^I(i) = f_i(s)[AID^I(p)]$
          end if
          if $collision = 1 \wedge AID^I(i) = nil$ **then**
            select $a \in LI^i(i) - SET$;
            create entry $ART^i[a, p, AID^I(p)]$; $AID^I(i) = f_i(v)[a]$
          end if
          $D^I(i) = D(i, n(j)^*, s)$;
          send $I[n(j), AID^I(i), D^I(i)]$ to s; **return**
        end if
      end for (% LFR is not satisfied; Interest may be traversing a loop)
      send $REP[n(j), AID^R(i), \text{loop}]$ to p
    end if
  end if
end if

Algorithm 2 shows the steps taken by router i to process an Interest received from a neighbor router p. The main differences in the steps taken by router i compared to Interests received from local users are that no Interest aggregation is done for Interests received from neighbor routers, and router i maps the AID it receives in the Interest from the previous hop to the AID it should use in the Interest it sends to the next hop using a simple mapping function.

When router i forwards Interest $I[n(j), AID^I(p), D^I(p)]$ from predecessor router p to successor router n towards name prefix $n(j)^*$, router i makes sure that $AID^I(p) \in LI^i(i)$ is not listed in an ART^i entry with a next hop other than p. If that is the case, router i stores $ART^i[AID^I(p), p, AID^I(p)]$, and sets $AID^I(i) = f_i(n)[AID^I(p)] \in LI^i(n)$. Otherwise, router selects an AID $b \in LI^i(n)$ that is not used to denote any other source of Interests in ART^i, stores $ART^i[b, p, AID^I(p)]$, and sets $AID^I(i) = f_i(n)[b] \in LI^i(n)$.

If the requested content is cached locally, a data packet is sent back. If router i is an origin of $n(j)^*$ and the CO with name $n(j)$ is not found locally, a reply is sent back stating that the content could not be found. Additional steps can be taken to address the case of malicious Interests requesting non-existing content. If the CO is remote and no FIB entry exists for $n(j)^*$, then router sends a reply stating that no route could be found for the CO.

Router i tries to forward the Interest to a next hop s for the best prefix match for $n(j)$ that satisfies LFR. The highest-ranked router satisfying LFR is selected as the successor for the Interest and router i. If no neighbor is found that satisfies LFR, a reply is sent stating that a loop was found.

Algorithm 3 Processing data packet from router s at router i

function Data Packet
INPUT: $LIGHT^i$, $LIST^i$, ART^i, $DP[n(j), AID^R(s), sp(j)]$;
[o] verify $sp(j)$;
[o] if verification with $sp(j)$ fails **then** discard $DP[n(j), AID^R(s), sp(j)]$;
$a = f_i^{-1}(s)[AID^R(s)]$; retrieve entry $ART^i[a, p, m]$;
if $ART^i[a, p, m]$ does not exist **then** drop $DP[n(j), AID^R(s), sp(j)]$;
if $p = i$ (% router i was the origin of the Interest) **then**
  for each $c \in lc[n(j)]$ **do**
    send $DP[n(j), c, sp(j)]$ to c; $lc[n(j)] = lc[n(j)] - \{c\}$
  end for
else
  if $p \in N^i$ **then**
    $AID^R(i) = m$; send $DP[n(j), AID^R(i), sp(j)]$ to p
  end if
end if
if no entry for $n(j)$ exists in $LIGHT^i$ **then**
  create $LIGHT^i$ entry for $n(j)$: $lc[n(j)] = \emptyset$
end if
store CO in local storage; $p[n(j)]$ = address of CO in local storage

Algorithm 3 outlines the processing of data packets. If local consumers requested the content in the data packet, it is sent to those consumers based on the information stored in $LIGHT^i$. If the data packet is received in response to an Interest that was forwarded from router p, router i forwards the data packet doing the proper mapping of AIDs. Router i stores the data object if edge or on-path caching is supported.

When router i receives $DP[n(j), AID^R(n), sp(j)]$ from neighbor n, it obtains the AID of of the destination where the packet should be forwarded by computing $f_i^{-1}(n)[AID^R(n)]$. Router i uses entry $ART^i[f_i^{-1}(n)[AID^R(n)], p, m]$ to determine the next-hop neighbor p that should receive the data packet, and sets $AID^R(i) = m$.

Algorithm 4 Process reply from router s at router i

function REPLY
INPUT: $LIGHT^i$, $LIST^i$, ART^i, $REP[n(j), AID^R(s), \text{CODE}]$;
$a = f_i^{-1}(s)[AID^R(s)]$; retrieve entry $ART^i[a, p, m]$;
if $ART^i[a, p, m]$ does not exist **then** drop $REP[n(j), AID^R(s), \text{CODE}]$;
if $p = i$ (% router i was the origin of the Interest) **then**
  for each $c \in lc[n(j)]$ **do**
    send $REP[n(j), c, \text{CODE}]$ to c
  end for
  delete entry for $n(j)$ in $LIGHT^i$
else
  if $p \in N^i$ **then**
    $AID^R(i) = m$; send $REP[n(j), AID^R(i), \text{CODE}]$ to p
  end if
end if

Algorithm 4 states the steps taken to handle replies, which are similar to the forwarding steps taken after receiving a data packet. Router i forwards the reply to local consumers if it was the origin of the Interest, or to a neighbor router p if it has an ART entry with p as the next hop towards the destination denoted by the AID stated in the reply.

D. Example

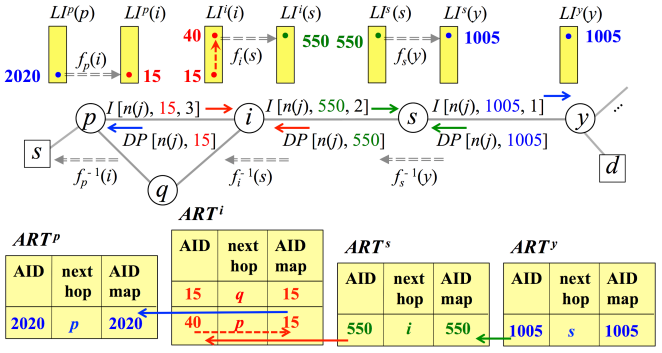


Fig. 3. Forwarding of Interests and responses to them in CCN-GRAM

Figure 3 illustrates the swapping of AIDs used by routers to forward Interests and responses to them. The local intervals used in the figure are small for simplicity, and the figure focuses on the forwarding state needed to forward Interests from p to name prefixes announced by router y, as well as the responses to such Interests. Interests are forwarded based on FIB entries, and responses to Interests (data packets or replies) are forwarded based on ART entries.

As illustrated in Figure 3, router i takes into account the possibility of collisions in the AIDs stated in Interests received from different neighbors by means of the identifier-mapping filed of ART entries. The bijection in Eq. 2 is used to map either the AID specified in the Interest received from neighbor p or the AID created by router i to handle collisions to the AID stated by router i in the Interest it forwards to a next-hop router s. In the example, router i has an exiting entry $ART^i[15, q, 15]$ when it receives Interest $I[n(j), 15, 3]$ from router $p \neq q$. Accordingly, router i selects $AID = 40$, creates entry $ART^i[40, p, 15]$, and sets $AID^I(i) = f_i(s)[40] = 550$ before forwarding Interest $I[n(j), 550, 2]$ to router s. When

router i receives data packet $DP[n(j), 550, sp(j)]$ from router s, it computes $f_i^{-1}(s)[550] = 40$. Using $AID = 40$ as the key in ART^i, router i obtains the next hop p, sets $AID^R(i) = 15$, and forwards $DP[n(j), 15, sp(j)]$ to router p.

It is clear from the example that a router sending an Interest is unaware of collisions of AIDs at the next hop. The identifier mapping field of ARTs allows routers to multiplex Interests from different neighbors stating the same AID values.

Even when a very small number of routers is involved, only the router that originates an Interest is able to determine that fact, because the identifiers used for Interest forwarding are assigned by the next hops.

E. Native Support for Multicasting

Support of multicast communication in the data plane with no additional signaling required in the control plane is viewed as an important benefit derived from maintaining per-Interest forwarding state using PITs. In short, multicast receivers send Interests towards the multicast source. As Interests from receivers are aggregated in the PITs on their way to the multicast source, a multicast forwarding tree (MFT) is formed and maintained in the data plane. Multicast Interest are forwarded using the same FIB entries used for unicast traffic, and multicast data packets are sent using reverse path forwarding (RPF) over the paths traversed by aggregated Interests. Using PITs is appealing in this context; however, as we show below, native support of multicasting in the data plane can be easily done with no need for per-Interest forwarding state!

1) Information Stored and Exchanged: We assume that the name stated in an Interest created to request content from a multicast source denotes a multicast source uniquely, and call such an Interest a *multicast Interest*. We also assume that consumers and routers differentiate between a multicast Interest and an Interest originated from a single consumer (unicast Interest).

A multicast Interest $MI[g(j), D^I(i), mc^I(i)]$ sent by router i to router n states: the name of a multicast group $g(j)$, the distance from router i to the source of the multicast group $D^I(i)$, and a multicast counter ($mc^I(i)$) used for pacing.

A multicast data packet $MP[g(j), sp(j), mc^R(i)]$ states the name of the multicast group $g(j)$, a security payload $sp(j)$, a multicast counter $mc^R(i)$, plus the content payload. A multicast reply $MR[g(j), CODE, mc^R(i)]$ states the reason for the reply and the current value of the multicast counter.

Router i maintains a multicast anonymous routing table ($MART^i$) that contains the forwarding state to the receivers of multicast groups. Each entry in $MART^i$ specifies a multicast group name, the value of the multicast counter (mc), and a list of next hops to the group of receivers who have sent Interests for the group. If router i has local receivers for group $g(j)$, the entry for the group in $MART^i$ includes router i as a next hop to the receivers of the group.

Router i also maintains a group membership table (GMT^i) that lists the mappings of multicast group names to the lists of local receivers that requested to join the groups. The GMT entries allow the router to deliver multicast content to local receivers of specific groups.

2) Multicast Content Dissemination: The key difference of the way in which CCN-GRAM forwards multicast traffic compared to NDN or CCNx is that a MART maintains per-group forwarding state, while a PIT maintains per-Interest forwarding state. Figure 4 illustrates the forwarding of multicast Interests and multicast content in CCN-GRAM. There is no need for anonymous identifiers for multicast content forwarding, because all consumers of a group must receive the same multicast COs, which are forwarded using multicast group names.

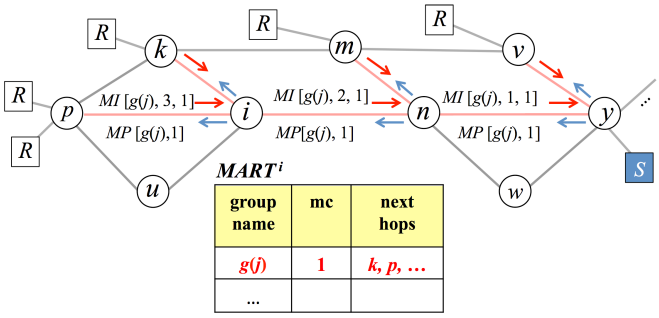


Fig. 4. Native multicast support in CCN-GRAM

A content consumer c requests to join a multicast group $g(j)$ as a receiver by sending a multicast Interest $MI[g(j), D^I(c), mc^I(c)]$ with $D^I(c) = nil$.

If router i has multiple local receivers or neighbor routers requesting to join the same multicast group $g(j)$, router i forwards multicast Interest $MI[g(j), D^I(i), mc^I(i)]$ only once towards the source of the multicast group $g(j)$ based on the information in its FIB. Router i simply adds new local consumers to the entry for $g(j)$ in GMT^i or new next hops to multicast receivers in $MART^i$.

Router i forwards multicast data packets based on the group names stated in the packets and the next hop stored in its MART entries, and discards the data packet if no MART entry exists for the multicast group. A similar approach is used for replies to Interests regarding multicast groups.

The dissemination of multicast data packets over the MFT of a multicast group can be of two types. A multicast source can push multicast data towards the receivers, or the receivers can pull data from the source by submitting Interests.

Push-based dissemination: The only forwarding state needed in CCN-GRAM for push-based multicast dissemination consists of the name of a multicast group and the names of the next hops towards the group receivers. In this mode, the mc value of an entry in a MART is updated with each multicast data packet forwarded by the router towards the receivers.

Pull-based dissemination: CCN-GRAM can also support pull-based multicast dissemination with no need for per-Interest forwarding state. An exemplary approach consists of a source-pacing algorithm based on the mc values carried in Interests and data packets. Each receiver increments the mc

value of Interests it sends for the group asking for the next piece of multicast content from the source. When router i receives multicast Interest $MI[g(j), D^I(p), mc^I(p)]$ from a neighbor router or a local content consumer p, it forwards the Interest only if $mc = 1 + v$, where v is the current mc value stored in $MART^i$ for the multicast group. Router i updates the mc value in $MART^i$ as it forwards the Interest, and subsequent Interests with the same mc value of $1 + v$ are simply dropped. As a result, each router in an MFT forwards a single copy of any Interest asking for the next multicast content object towards the source. This is like aggregating Interests for a multicast group over the MFT of the group, but with no need to store per-Interest forwarding state.

IV. Performance Comparison

We compare the forwarding entries needed to forward Interests and responses in NDN and CCN-GRAM, as well as the end-to-end delays incurred, using simulation experiments based on implementations of NDN and CCN-GRAM in the ndnSIM simulation tool [1]. The NDN implementation was used without modifications, and CCN-GRAM was implemented in the ndnSIM tool following Algorithms 1 to 4.

The network topology consists of 150 routers distributed uniformly in a 100m $\times$ 100m area and routers with distance of 15m or less are connected with point-to-point links of delay 15ms. The data rates of the links are set to 1Gbps to eliminate the effects that a sub-optimal implementation of CCN-GRAM or NDN may have on the results. Only 10 routers chosen randomly are connected to local content producers of multiple name prefixes, 50 other routers are connected to local content consumers, and all routers act as relays. This choice was made to illustrate the existence of a "network edge" and the fact that only a relatively small number of sites host content producers. Interests are generated with a Zipf distribution with parameter $\alpha = 0.7$ and producers are assumed to publish 1,000,000 different COs. Each cache can store up to 1000 objects, or 0.1% of the content published in the network. This caching capacity was selected to compare on-path caching with edge caching when Interests must be forwarded in the network, rather than being answered with locally cached content.

We considered *total Interest rates per router* of 50, 100, 500, and 2000 objects per second corresponding to the sum of Interests from the local consumers connected to a router. The increasing values of total request rates can be viewed as higher request rates from a constant user population of local active users per router, or an increasing population of active users per router. The Interest rates we assume are actually very low according to recent results addressing the size that PITs would have under realistic Internet settings [8], [22], [23], [21].

We considered on-path caching and edge caching. For the case of on-path caching, every router on the path traversed by a data packet from the producer to the consumer caches the CO in its local cache. On the other hand, with edge caching, only the router directly connected to the requesting consumer caches the resulting CO. All caches are LRU.

A. Size of Forwarding Tables

Figure 5 shows the average size and standard deviation of the sizes of PITs, ARTs and LIGHTs on a logarithmic scale as functions of Interest rates. The size of LIGHTs corresponds only to the number of local Interests pending responses. The number of entries corresponding to content cached locally can be up to 1000 for both NDN and CCN-GRAM.

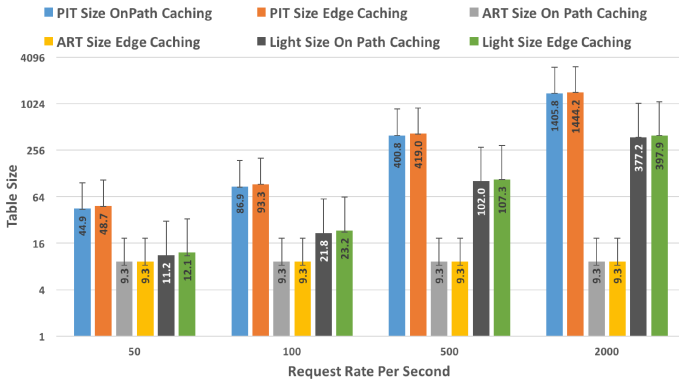


Fig. 5. Average size of forwarding tables

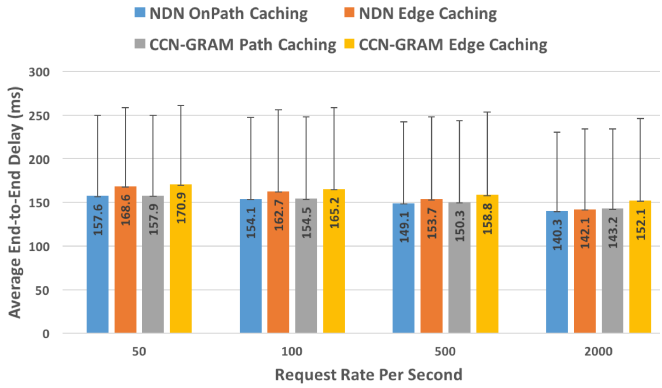


Fig. 6. Average end-to-end delays

As the figure shows, the size of PITs grows dramatically as the rate of content requests increases, which is expected given that PITs maintain per-Interest forwarding state. By contrast, the size of ARTs, which is the only forwarding state stored by relay routers, is only a small fraction of the total number of routers and remains fairly constant with respect to the content request rates, which is always one or multiple orders of magnitude smaller than the average PIT size. The size of LIGHTs is a function of the number of COs requested locally or cached on path, but the average size of a LIGHT is an order of magnitude smaller than the average size of a PIT. The size of a ART is independent of where content is being cached, given that an ART entry is stored independently of how many Interests traverse the route. Interestingly, edge-caching renders only slightly larger PIT sizes than on-path caching in NDN.

B. Average Delays

Figure 6 shows the average end-to-end delay for NDN and CCN-GRAM as a function of content request rates for on-path caching and edge caching. As the figure shows, the

average delays for NDN and CCN-GRAM are comparable for all values of the content request rates. This should be expected, given that the static, loop-free routes in the FIBs prevent Interests to "wait to infinity" in PITs, the signaling overhead incurred by NDN and CCN-GRAM is similar, and in-network caching obviates the need for Interest aggregation.

V. Conclusions and Future Work

We presented simulation results showing that Interest aggregation rarely occurs when in-network caching is used. Our analysis is limited; however, our detailed characterization of Interest aggregation via analytical modeling and simulation analysis [7] renders the same conclusion.

We introduced CCN-GRAM to eliminate the performance limitations associated with PITs. CCN-GRAM is the first approach to Interest-based content-centric networking that supports the forwarding of Interests and responses to them using datagrams that do not reveal the identity of their origins to forwarding routers, caching sites, or content providers.

Simulation experiments were used to show that end-to-end delays incurred in CCN-GRAM and NDN are similar when either edge caching or on-path caching is used, but the storage requirements for CCN-GRAM are orders of magnitude smaller than for NDN. The results for CCN-GRAM indicate that it could be deployed with only routers at the edge maintaining LIGHTs and caches. Additional work is needed to make the forwarding of Interests in CCN-GRAM as efficient as the forwarding of responses to Interests using ARTs. The goal is to enable Interest forwarding at Internet scale that does not require routers to look up FIBs with billions of name-prefix entries as is the case in NDN and CCNx.

Both ARTs and PITs must be updated when the paths traversed by Interests and their responses must change due to congestion, topology changes, or mobility of consumers and providers. Yi et al [27] argue that per-Interest forwarding state enables faster response to topology changes and congestion, because local repair mechanisms can be used. However, multipath routing, and dynamic load balancing schemes based on datagram forwarding have been shown to attain results very close to optimal routing [24] and can be easily applied to CCN-GRAM in the future.

CCN-GRAM can use the same content security features adopted in CCNx and NDN to limit or eliminate cache poisoning attacks, because it makes no modifications to the way in which content is protected in data packets or how a name can be securely linked to the payload of a CO. However, CCN-GRAM enjoys an enormous advantage over CCNx and NDN in that it eliminates the ability for malicious users to mount Interest-flooding attacks aimed at overwhelming the forwarding tables of routers [16], [23]. An ART entry can be added only for valid local identifiers at each router and for routes that satisfy the ordering constraint imposed with LFR. Given that both conditions are managed in the control plane, mounting attacks on ARTs is much more difficult than simply having users send Interests for COs corresponding to valid name prefixes.

References

[1] A. Afanasyev et al., "ndnSIM: NDN simulator for ns-3", *University of California, Los Angeles, Tech. Rep*, 2012.

[2] A. Afanasyev et al., "Interest-flooding Attack and Countermeasures in Named Data Networking," *Proc. IFIP Networking '13*, May 2013.

[3] AT&T, "The Quality of Internet Service: AT&T's Global IP Network Performance Measurements," 2003. http://ipnetwork.bgtmo.ip.att.net/pws/paper.pdf

[4] P. Baran, "On Distributed Communications: I. Introduction fo Distributed Communication Networks," Memorandum RM-3420-PR, The RAND Corporation, Aug. 1964.

[5] Content Centric Networking Project (CCN) [online]. http://www.ccnx.org/releases/latest/doc/technical/

[6] A. Dabirmoghaddam et al., "Understanding Optimal Caching and Opportunistic Caching at The Edge of Information Centric Networks," *Proc. ACM ICN '14*, Sept. 2014.

[7] A. Dabirmoghaddam, M. Dehghan, and J.J. Garcia-Luna-Aceves, "Characterizing Interest Aggregation in Content-Centric Networks," *Proc. IFIP Networking 2016*, May 2016.

[8] H. Dai el al., "On Pending Interest Table in Named Data Networking," *Proc. ACM ANCS '12*, Oct. 2012.

[9] C. Fricker et al., "Impact of traffic mix on caching performance in a content-centric network," *Proc. IEEE NOMEN Workshop '12*, 2012.

[10] J.J. Garcia-Luna-Aceves, "Name-Based Content Routing in Information Centric Networks Using Distance Information," *Proc. ACM ICN '14*, Sept. 2014.

[11] J.J. Garcia-Luna-Aceves, "A Fault-Tolerant Forwarding Strategy for Interest-based Information Centric Networks," *Proc. IFIP Networking '15*, May 2015.

[12] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "Enabling Correct Interest Forwarding and Retransmissions in a Content Centric Network," *Proc. ACM ANCS '15*, May 2015.

[13] J.J. Garcia-Luna-Aceves, "A More Scalable Approach to Content Centric Networking," *Proc. IEEE ICCCN '15*, Aug. 3-6, 2015.

[14] J.J. Garcia-Luna-Aceves, "Eliminating Undetected Interest Looping in Content Centric Networks," *Proc. IEEE NOF '15*, Sept. 30-Oct. 2, 2015.

[15] J.J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "A Light-Weight Forwarding Plane for Content Centric Networks," *Proc. IEEE ICNC '16*, Feb. 2016.

[16] P. Gasti et al., "DoS and DDoS in Named Data Networking," *Proc. IEEE ICCCN '13*, 2013.

[17] A.K.M. Mahmudul-Hoque et al., "NSLR: Named-Data Link State Routing Protocol," *Proc. ACM ICN '13*, 2013.

[18] NDN Project [online]. http://www.named-data.net/

[19] J. Raju et al., "System and Method for Information Object Routing in Computer Networks," U.S. Patent 7,552,233, June 23, 2009

[20] M. Spohn and J.J. Garcia-Luna-Aceves, "Neighborhood Aware Source Routing," *Proc. ACM MobiHoc 2001*, Oct. 2001.

[21] C. Tsilopoulos et al., "Reducing Forwarding State in Content-Centric Networks with Semi-Stateless Forwarding," *Proc. IEEE INFOCOM '14*, April 2014.

[22] M. Varvello et al., "On The Design and Implementation of a Wire-Speed Pending Interest Table," *Proc. IEEE Infocom NOMEN Workshop '13*, April 2013.

[23] M. Virgilio et al., "PIT Overload Analysis in Content Centric Networks," *Proc. ACM ICN '13*, Aug. 2013.

[24] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Simple Approximation to Minimum-Delay Routing," *Proc. ACM SIGCOMM '99*, Aug. 1999.

[25] M. Wahlisch et al., "Lessons from the Past: Why Data-driven States Harm Future Information-Centric Networking," *IFIP Networking '13*, May 2013.

[26] M. Wahlisch et al., "Backscatter from The Data Plane? Threats to Stability and Security in Information-Centric Network Infrastructure," *Computer Networks*, Vol. 57, No. 16, Nov. 2013.

[27] C. Yi et al., "A Case for Stateful Forwarding Plane," *Computer Communications*, pp. 779-791, 2013.

[28] L. Zhang et al., "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, Vol. 44, No. 3, July 2014.

BEAD: Best Effort Autonomous Deletion in Content-Centric Networking

Cesar Ghali* Gene Tsudik* Christopher A. Wood+
University of California, Irvine
Email: {cghali, gene.tsudik, woodc1}@uci.edu

***Abstract*—A core feature of Content-Centric Networking (CCN) is opportunistic content caching in routers. It enables routers to satisfy content requests with in-network cached copies, thereby reducing bandwidth utilization, decreasing congestion, and improving overall content retrieval latency.**

One major drawback of in-network caching is that content producers have no knowledge about where their content is stored. This is problematic if a producer wishes to delete its content. In this paper, we show how to address this problem with a protocol called BEAD (Best-Effort Autonomous Deletion). It performs content deletion via small and secure packets that resemble current CCN messages. We discuss several methods of routing BEAD packets from producers to caching routers with varying levels of network overhead and efficacy. We assess BEAD's performance via simulations and provide a detailed analysis of its properties.

Keywords*—*Content-Centric Networking, Information-Centric Networking, caching, best-effort content deletion, controlled flooding, forwarding histories, accounting.

I. Introduction

Content-Centric Networking (CCN) is a relatively recent internetworking paradigm touted as an alternative current IP-based Internet architecture. While IP traffic consists of packets between communicating end-points, CCN traffic is comprised of explicit requests for, and responses to, named content objects.

An important feature of name-based content retrieval is decoupling of content from its producer. This enables more natural content distribution by allowing routers to opportunistically cache content *within the network*. Cached content can be returned in response to future requests, called *interests*. This reduces the need to forward interests to content producers, thus lowering network congestion and content retrieval latency.

However, router caches are not mandatory in CCN. In some cases, caching content might not be beneficial, e.g., for routers with high content processing speeds, since high arrival rates translate to less time spent in cache. If the content's cache lifetime is very short, the probability of cache misses increases and the cache's utility decreases commensurately. Indeed, some prior literature shows (via simulations and experiments) that caching at the edges of the internetwork, i.e., at consumer-facing routers, is most beneficial and more cost-effective than doing so in the core, i.e., in transit routers [1].

*Supported by NSF award: "CNS-1040802: FIA: Collaborative Research: Named Data Networking (NDN)".

+Supported by NSF Graduate Research Fellowship DGE-1321846.

ISBN 978-3-901882-83-8 © 2016 IFIP

To help caching routers determine the lifetime of cached content, the latter includes an optional `ExpiryTime` field. Routers are expected to flush content once this time elapses. However, a router can choose to keep content cached beyond its lifetime. Lifetime of content in a particular router's cache depends entirely upon that router's implementation and policy. This uncertainty (or freedom) means that content may linger in the network for a very long time.

One notable drawback of this libertarian approach to caching is that some content may need to be deleted *before* `ExpiryTime` elapses. Consider content that frequently (yet sporadically) evolves over time, e.g., news articles. The appearance of breaking-news articles is unscheduled. As situations develop, updates and corrections to the content occur at unpredictable times. Such updates supersede previously distributed content by rendering it stale. Thus, in this case, producers need a way to remove old content. Another example is content (that has released and subsequently cached) which contains erroneous information. As errors are detected and corrected, a producer needs to flush the incorrect older version.

The deletion problem occurs because `ExpiryTime` is the only way for a producer to communicate *anticipated* content lifetime to the network. However, a producer can not change its mind after content has been published and distributed. Thus, there is a need for a safety mechanism for in-network content deletion. For this reason, we design such a technique called BEAD: Best-Effort and Autonomous Deletion. In the process, we encounter and address several challenges, including efficacy, performance, and security. We also experimentally assess the proposed technique.

The rest of this paper is organized as follows. Section II overviews CCN. Related work is summarized in Section III. Section IV presents minimal requirements for content deletion. Sections V and VI describe authentication and routing of deletion requests in BEAD, respectively. The BEAD technique is analyzed in Section VII and its performance is assessed in Section VIII. The paper ends with a discussion of BEAD optimizations and practical factors in Section IX. Future work is summarized in Section X.

II. CCN Overview

We now summarize the current CCN architecture [2]. Given familiarity with CCN, it can be skipped without loss of continuity.

Unlike IP, which focuses on addressable end-hosts, CCN emphasizes named and addressable content. A consumer issues a request, called an *interest*, specifying the name of desired

content. CCN names are structured similar to URIs. For example, a content produced by the NSA might be named: `ccnx:/us/gov/DoD/NSA/Snowden-Diary`. An interest for a particular content named N is routed towards an authoritative producer for that content, based on N itself. In CCN, both interest and content messages have general-purpose `Payload` fields. Consumers can use an interest's `Payload` field to *push* information to producers, while producers use a content's `Payload` field to carry actual application data.

As an interest traverses the network, each router determines if a copy of requested content is cached in its Content Store (CS). If a cache hit occurs, the router satisfies the interest by sending the matching content on the interface on which the interest arrived. Otherwise, the router (1) records some state derived from the interest in its Pending Interest Table (PIT) in order to provide a backwards path for the future content, and (2) forwards the interest to the next hop(s) specified in its Forwarding Information Base (FIB). State retained in the PIT contains the content name and the interface(s) on which interests for that name have been received. A FIB is a routing table that maps hierarchical name prefixes to outbound interfaces. Longest-Prefix Matching (LPM) is used to determine the matching FIB entry.

A router R can collapse multiple interests into the same PIT entry whenever all of the following holds:

1) R receives an interest for name N
2) R does not have content N in its cache
3) R's PIT already contains an entry for N

When interest collapsing occurs, R only records the interface on which the new interest arrived and drops that interest. Whenever requested content arrives, R forwards it on all interfaces listed in the corresponding PIT entry. Afterwards, the PIT entry is flushed.

If no router can find a locally cached copy of requested content, the interest eventually reaches the producer that responds with the matching content, if possible. If the producer can not provide it (e.g., content does not exist) a NACK is generated [2], [3]. As content traverses the reverse path to the consumer, routers may choose to cache it in anticipation of future requests. As mentioned earlier, each content includes a producer-set `ExpiryTime` field. This value is content- and application-specific. However, each router can use any cache management algorithm, e.g., LRU or LFU.

III. Related Work

Lack of on-demand content deletion is a well-known problem in CCN [4]–[9]. The problem of *unsafe replicas* or stale content in CCN was first considered in [10]. Analytical and experimental assessment showed that: *"...the more frequently content is requested the higher is the chance of one request ending up in between a revocation and the eviction [of the stale key]."* The proposed method relies on a monotonically decreasing cache lifetime enforced by cooperating routers. This does not allow a producer to change the lifetime after content is published; it only seeks to minimize the time window when stale or unsafe replicas can be accessed.

[4] proposed a mechanism to implement revocation of content without input from the consumer. The proposed approach uses the ccnx-sync protocol to perform OCSP-like [11] synchronization of key data, i.e., determine content that has been revoked. This requires proactive behavior by each participating repository. [5] suggests using ChronoSync [12] to synchronize revoked key endorsements among group members. Revocation, however, is not the same as cache deletion. Revoked content, if still cached, can be inadvertently accessed by malicious or benign consumers.

[13] discussed a new caching technique allowing routers to proactively share content with downstream peers which did explicitly request that content. The suggested multicast forwarding strategy serves to increase the number of replicas in the network. However, unsolicited content objects can be seen as a form of attack similar to cache poisoning [7].

The concept of cost-aware caching in CCN was introduced in [14]–[18]. Various economic incentives for ISPs and ASs to cache content on behalf of producers have been explored. Cost-aware routers that cache based on popularity and economic incentives are studied in [19]. In general, the economic problem of supporting prioritized caching in the network is addressed without any attention to the inverse problem: how is content removed from caches?

IV. BEAD Requirements

Our motivation stems from the need to remove stale or erroneous content from the network, i.e., from routers' caches. One intuitive way of doing this is through the use of versioning, whereby the content naming format includes a component that explicitly reflects the current version. For example, the content of BBC's World News web-page could be named: `ccnx:/bbc/news/world/v2.4`. Alternatively, timestamps could be used. In that case, the same BBC page could be named `ccnx:/bbc/news/world/1449187200`.[1] However, in either case, is unclear how a consumer would determine (in advance) the current timestamp or version number, without which an interest can not be formed.[2]

The main problem with versioning and timestamps is that they can not handle unpredictable content updates. In current CCN design, producers are oblivious to where and for how long their content is stored in the network. Although this opportunistic caching is one of the biggest CCN advantages, it greatly complicates deletion of stale content. We believe that, in order to address the problem, producers need:

1) A way to communicate a single deletion request to all routers that might have cached offending content.
2) A way to efficiently secure deletion requests (allowing routers to quickly authenticate them) while avoiding trivial Denial of Service (DoS) attacks.

The first requirement is reminiscent of IP traceback – a class of techniques for identifying the original source of a (usually malicious) packets. In the context of IP, this is often framed as a mechanism to mitigate Denial of Service (DoS) attacks. In this paper, the goal is to learn *where* content was previously forwarded so that deletion requests can be routed along the same paths. These paths correspond to the original sources

[1] 1449187200 is 12/04/2015 at 12:00am UTC.

[2] There is one trivial way: a consumer contacts the producer directly and asks for the most recent version number or timestamp. However, this would incur an extra round-trip delay per content retrieval.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 181

of interests for that content. Thus, ideas from IP traceback based on packet logging (e.g., [20]) and (deterministic or probabilistic) packet marking (e.g., [21], [22]) influence the design and forwarding strategies of BEAD messages.

We now show how to address these requirements with the BEAD technique.

V. Authenticating Deletion Requests

Producers must prove content ownership to routers that receive deletion requests. Otherwise, an adversary can impersonate a producer and induce content deletion, resulting in another form of DoS. One way to attain authentication is by a producer-generated digital signature on each deletion request. However, besides being inefficient, forcing routers to verify signatures on deletion requests can be itself parlayed into DoS attacks [7], [23]. Moreover, it involves public key retrieval, certificate handling and other messy (for routers) issues.

Our approach uses a light-weight token that proves content ownership. When a producer P creates a content object C, it generates a random λ-bit string x_C, called the *deletion token*. P then computes the digest of this token using a suitable cryptographic hash function[3], $y_C = H(x_C)$, and includes y_C in C. Later, if and when P wishes to delete C from the network, it includes x_C in the deletion request. (We assume that P can route these requests to any router caching C.) Upon receipt, each R verifies that y_C (cached alongside the content) matches $H(x_C)$. If so, R knows that P must have issued the request and deletes C from the cache.[4]

VI. Routing Deletion Requests

The remaining (though major) issue is how to route deletion requests from the producer to each caching router. This can be viewed as a multicast problem where producers must distribute a message (deletion request) to only a subset of nodes which could have cached the content.

Let $Int[N]$ and $C[N]$ be the interest and content messages referring to name N. The hash of $C[N]$ is a λ-bit string d, i.e., $d = H(C[N])$. Let $E[N, d]$ be a deletion request for content named N and hash digest d. Let $\mathbb{R}_N$ be the set of routers which cached $C[N]$. Finally, let the FIB of router $R \in \mathbb{R}_N$ be FIB^R.

From here on, we use the term erase to refer to deletion requests. Also, we assume that erase messages are authenticated using the method described in Section V.

A. Flooding

We begin by considering the simplest approach: reverse-path controlled flooding [24] of deletion requests. When $R \in \mathbb{R}_N$ receives $E[N, d]$, it forwards it on all interfaces except those which have a matching FIB entry.

Flooding offers some advantages, the most important of which is the ability to reach network edges even if routers on the producer-to-consumers paths no longer cache the content to be deleted. This is important since routers do not cache content uniformly and some may not even have caches. On the negative side, the volume of traffic generated from a single deletion request is very high and most deletion requests would be forwarded to routers that never even cached the target content.

B. Forwarder Histories for Content Traceback

In the optimal case, routers would only forward erase messages on interfaces on which the referenced content had been previously forwarded. In other words, erase messages should only be forwarded along the content distribution spanning tree where the producer is the root and leaves are the consumers who requested the content. One way to forward erase messages along the edges of this tree is for each router $R \in \mathbb{R}_N$ to maintain a forwarding history of $C[N]$. There are several places where this history can be kept, including: (1) in the cache where $C[N]$ is stored, (2) in a forwarding log (similar to [20], as a form of IP traceback) at each router, and (3) in the packets themselves. In each case, historical information constitutes a form of traceback that allows routers to identify where content was previously forwarded. We now describe each approach in more detail.

1) In-Cache Forwarding Histories: When a router caches $C[N]$ it can also remember the downstream interfaces where the cached copy was forwarded. We denote the set of these interfaces as $\mathbb{F}_N$. When a router receives an interest $Int[N]$ on interface F_i, it responds with $C[N]$ and adds F_i to $\mathbb{F}_N$. For a router with K interfaces, this additional state costs $\mathcal{O}(K)$ bits per cache entry. When a router caching $C[N]$ receives $E[N, d]$, it forwards it on all interfaces in $\mathbb{F}_N$.

In-cache forwarding histories are only effective for routers with large caches, since the lifetime of forwarding information is bound to the lifetime of cache entries, which can be small or even zero (if a router has no cache at all). Since a forwarding history $\mathbb{F}_N$ is deleted whenever $C[N]$ is flushed from the cache, this can lead to a future $E[N, d]$ not being forwarded to downstream routers which might still cache $C[N]$.

2) Local Forwarding Logs: Long-term packet logs have their roots in IP traceback techniques from the early 2000-s, e.g., [20], [25]. The problem here is similar: routers need long-term histories of packets (content) that were previously processed and forwarded. In this context, a history is a set-like data structure that allows content objects to be inserted and then later queried for membership. There are two types of histories: *lossless* and *lossy*. The former always return "yes" for content objects that have previously been inserted. In contrast, a lossy history might return false positives or negatives. Routers use these structures by associating one history to each interface. When a router receives $E[N, d]$ and $C[N]$ is not cached, it forwards $E[N, d]$ on each interface for which the corresponding forwarding interface history has a record of $C[N]$, i.e., all histories for which membership query returns "yes". This procedure is outlined in Figure 1.[5]

We now describe some ways of implement lossless and lossy histories that vary in their computation and memory requirements.

Lossless Forwarder Histories require a unique identifier to be kept after a content object has been forwarded. We

[3] Suitable hash functions include those with pre-image resistance, which means that, given y, it is difficult to find an x such that $y = H(x)$.

[4] This is due to the randomness of x_C and the collision-resistance of $H(\cdot)$.

[5] Similar to the flooding algorithm, this check is not performed for interfaces via which the content producer can be reached.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 182

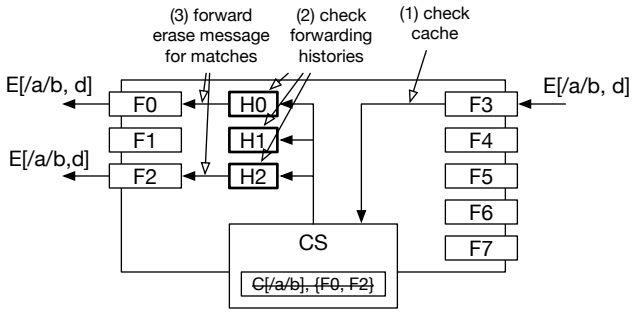


Fig. 1. $E[N, d]$ forwarding strategy based on per-interface forwarding histories. Upon receipt of an erase message a router searches the cache for the respective content. If the content is not present then the histories are examined and the history is forwarded as needed.

assume that content hash d serves as such an identifier (with collision probability negligible in λ). Implementing this type of forwarder history can be done trivially with a hash set HS_R as follows: to insert a content object into the history, compute and store d in HS_R. To query the history, return "yes" if $d \in HS_R$ and "no" otherwise. Insertion and lookup each require constant time.

Lossy Forwarder Histories are intended to store historical information in memory-constrained systems at the cost of false positives and false negatives. Similar to SPIE traceback [20], we use Bloom Filters (BFs) [26] to implement lossy forwarder histories. BFs enable probabilistic set membership queries.

The choices of BF properties, e.g., size and hash functions, impact efficacy of this technique. Filters that saturate too quickly result in high false positive rates. If all interface filters become saturated then erase is effectively broadcast. Therefore, it is important to eventually remove stale elements from filters. Unfortunately, a regular BF does not provide element removal. However, so-called Counting Bloom Filters (CBFs) [27] support set membership queries with removal. Instead of using bits to indicate set membership, CBFs use counters. When loading an element into CBF, the counters corresponding to the output of the hash functions are increased by one. Consequently, removing an element is done by decrementing the same counters. The problem with CBFs is that one must know the element to delete. Since routers would discard content after inserting them into these filters[6], they have no way of knowing what content is in the filter, and thus what elements to eventually delete. Their only recourse is to remove elements by decrementing counters at random. Intuitively, a router would delete random elements from the filter (the history) at a frequency which reflects the average `ExpiryTime` of received content. This can increase the false negative probability and reduce the possibility of delivering erase messages to their corresponding destination.

Variants of the CBF, such as Time-Decaying (TDBFs) [28], [29] and Stable (SBFs) [30] BFs can also be used. TDBFs have the property that elements are slowly removed from the filter over time, thereby keeping the rate of false positives minimized. However, the natural decay property may lead to false negatives. SBFs on the other hand are dynamically self-resized to keep the false probabilities minimized. Similar to CBFs and TDBFs, SBFs also suffer from false negatives.

[6]This is because content is only added to histories upon its removal from the cache.

3) Interest Marking for Content Traceback: Packet marking is a standard technique for IP traceback [21]. In the context of this work, marking is performed on interests to indicate sources of content requests. This information can be later used to learn the interface to which an erase needs to be sent. Specifically, erase messages can carry this marking information in order for routers to identify the appropriate downstream interfaces without storing any local state.

One trivial marking method is to append the arrival interface to each interest. Specifically, when R receives $Int[N]$ on face F_i, R prepends (R, F_i) to a list contained in the header of the interest. Producers record these traces upon receipt. In the event that an erase needs to be generated, P includes the trace in the erase and forwards it on the appropriate downstream interface. When R receives an erase with a trace it pops the last element (R, F_i) off the trace list and forwards it on the specified interface F_i.

This technique distributes the forwarding history among messages in the network. Therefore, this information must be secure. To illustrate this requirement, assume router R_i receives $E[N, d]$ with the sequence of hops

$$[(R_i, F_i), (R_{i-1}, F_{i-1}), \ldots, (R_2, F_2), (R_1, F_1)]$$

from interface F_{i+1}. R_i needs a way to securely guarantee that (R_i, F_i) was previously prepended, by itself, to the subsequence:

$$[(R_{i-1}, F_{i-1}), \ldots, (R_2, F_2), (R_1, F_1)].$$

Otherwise, the adversary can forge unsolicited erase messages with apparently correct routing sequences. Alternatively, one can modify existing sequences in erase messages to prevent them from being routed towards their destination.

One way of authenticating hop-sequence traces is for R_i to compute a Message Authentication Code (MAC) [31], [32] tag t_i over the (relevant) interest details, e.g., the name and previously present traces in the hop-sequence. R_i then adds the tuple (R_i, F_i, t_i) to the interest before forwarding it. Since erase messages carry the name of the content to be deleted, each router will be able to verify its pre-computed tag before forwarding erase messages downstream. Since routers compute and verify tags locally, a key management and distribution protocol is not required. We do, however, assume that routers are able to generate and maintain cryptographic keys of sufficient length necessary for MAC computation. As an added feature, hop-sequence information can also be used for detecting both interest and erase loops [33].

Although this technique of marking interest is effective to deliver erase messages to all routers on the path between consumers and producers, it has several drawbacks. One of this is that interest traces received by producers need to be stored so that they can be included in erase messages. This is due to the fact that (1) each trace corresponds to only one path in the network, and (2) interests issued by multiple consumers are most likely to traverse different paths to the producer. Producers can attempt to compile all collected traces in a data structure forming a spanning tree. This structure would

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 183

be included in erase message headers, allowing routers to forwarder erase messages correctly. The main disadvantage of this approach is that the size of the data structure grows linearly with the number of consumers and is most likely to be greater than average allowed MTU. This means that erase messages will be fragmented (and possibly re-fragmented), and hop-by-hop reassembly is not avoidable. Another alternative is for producers to send multiple erase messages one for each set of traces correlated to a hop-sequence. In Section VII, we compare and evaluate the performance and resource consumption of these two techniques.

VII. Analysis

We now assess some routing strategies for erase messages. Let n_t^R be the total number of content objects forwarded by R at time t and let μ_F^R be R's content forwarding rate. Note that n_t^R grows monotonically as a function of μ_F^R.

A. Flooding Analysis

Recall that the reverse path flooding algorithm works by only sending broadcast messages to interfaces through which the *producer* is not reachable. Though very effective, this is highly unscalable. If all routers flooded erase messages then they would certainly be delivered to every $R \in \mathbb{R}_N$. However, the number of routers receiving a specific erase message is much larger than $|\mathbb{R}_N|$. Therefore, flooding should always be the last resort for erase messages. We assess the actual overhead of this technique in Section VIII.

B. Forwarding History Analysis

We now analyze performance of lossless and lossy forwarding histories described in Section VI-B.

1) Lossless Histories: The memory (and possibly computational) cost of a lossless forwarder history grows as a function of t. Thus, history collection will inevitably saturate memory at some point. Let n_{max}^R be the total size (in entries) of the history memory for R. Saturation is reached at time t such that $n_t^R \geq n_{max}^R$. We compute the time required to saturate a lossless forwarder history in two scenarios. We assume that each content object is $4,096$ bytes and hash digests are 32 bytes.

- **Consumer-facing router:** We assume a caching consumer-facing router (e.g., an access point) with 4GB of history storage and data rate of 100 Mbps. This data rate is equivalent to a content forwarding rate of $\mu_F^R = 3'200$ Cps (content packets per second). If R operates at full capacity with a full cache – i.e., storing every forwarded content requires eviction of an already cached one – it will take $41,943$ secs. for history storage to be saturated. This is roughly 12 hours. This window of time might be longer than the `ExpiryTime` of content objects that are subject to be erased. For instance, news feed pages are likely to be updated with a frequency faster than $1/12$ hours.
- **Core router:** We assume a non-caching CCN core router with 1TB of flash history storage and data rate of 10 Tbps, i.e., equivalent to $\mu_F^R = 335$ MCps. If R always operates at full capacity (i.e., forwards at 10 Tbps), lossless forwarder history can be saturated in 102 secs. In this case producers have a time window of less than 2 minutes to issue an erase message for content C after it was last served.

R's saturation time can be lengthened by increasing the size of the forwarder history. However, at this rate, the cost of adding more memory to make saturation time useful is far too expensive: 1TB for 2 minutes of history.

A very natural question arises: what happens when R's history storage is saturated? R can evict old history entries randomly, or according to some policy, e.g., LRU. However, keeping track of history entries' ages might lead to reduced performance. Another alternative is to divide history storage into smaller chunks, each corresponding to a set time window of history entries. Once history storage is saturated, the oldest chunk is erased to provide space for new entries. Using the consuming-facing router example above, 4GB of history storage can be divided into 12 chunks, each corresponding to one hour. The router could then erase the history recorded 12 hours ago in order to store history entries for the coming hour.

2) Lossy Histories: Lossy histories are useful when lossless ones are too expensive, e.g., in core network routers. Our approach to lossy forwarder history is based on Bloom Filters (BFs) – probabilistic data structures with tunable performance. Given an m-bit BF that stores n elements, the number of input hash functions k can be optimized and false positive probability can be estimated using Equation 1 [34]. Note that optimal value of k is also given as a function of m an n.

$$f(m, \cdot, n) \approx (0.6185)^{\frac{m}{n}}, \quad k = \ln(2) \cdot \frac{m}{n} \tag{1}$$

In practice, a router can optimize the number of hash functions in order to lower false positive probability. An upper bound of k can be set to limit hashing overhead.

As mentioned above, standard BFs do not support entry deletion, which is necessary to deal with the saturation problem. As indicated in [20], historical information for Internet-scale traffic (IP packets) can not last beyond a few minutes, which might still be less than what we needed for BEAD.

We now analyze lossy forwarder histories in the context of two scenarios mentioned above with the same history storage and data rates. We also assume that each content object added to a BF changes the value of new distinct k bits from 0 to 1. Clearly, this is unrealistic, since we do not consider the possibility of overlapping of hash function outputs for different input elements. However, this assumption captures the worst-case scenario.

- **Consumer-facing router:** To maintain a maximum false positive probability of 10^{-32}, a BF of size 4 GB can fit $n \leq 2 \times 10^8$ elements. Based on Equation 1, it requires $k = 120$ hash functions. Thus, it will take $89'478$ secs. (a little over one day) for the forwarder history to be saturated.
- **Core router:** To maintain the same false positive probability, a BF of size 1 TB can accommodate $n \leq 5.7 \times 10^8$ elements, which corresponds to $k = 107$ hashes. The forwarder history will be saturated in 245 secs.

One major drawback of using BFs for lossy forwarding histories is that history saturation is more difficult to resolve. Recall that, with lossless histories, a router can remove old entries in

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 184

order to add new ones. A router could also delete the oldest chunk of the history once it is saturated. However, with lossy histories, a router can either: (1) flush the entire lossy history and start over, or (2) use CBFs which support element deletion with the use of counters. Unfortunately, this introduces false negative probabilities.

3) Packet Marking Analysis: Packet marking is computationally inexpensive since it requires a single MAC computation per (either interest or erase) packet. However, its drawback is increased memory footprint of the interest along every hop. Recall that traces in the hop-sequence consist of: (1) router identifier, (2) interface identifier, and (3) tag. Assuming a 2-byte interface identifier and a SHA-256-based MAC, the total size of each trace is 38 bytes. This corresponds to extra 608 bytes for each interest, assuming a 16-hop router-level path.[7]

We now compare two hop-sequence techniques described in Section VI-B3. Assume a tree topology with: (1) producer P at the root with height h, (2) 2^h consumers at the leaves with height 0, and (3) $2^h - 2$ routers. We assume all consumers request content C and all routers append hop-sequence traces to the corresponding interests. In this case, P receives 2^h interests, each with $h-1$ traces. If P includes all these traces in a single erase message, its size would grow by $\left(2^h \cdot (h-1)\right) \times 38$ bytes. This grows to 35 MB for $h = 16$, which is clearly impractical.[8] On the other hand, if P decides to send a separate erase to each consumer it would generate 2^h erase messages. The same overall volume of traces (35 MB) will be sent from P to consumers. However, it would be split into numerous erase messages. One advantage is that erase messages size will likely not exceed the path MTU and therefore not require fragmentation.

C. Summary of BEAD

As follows from the above, BEAD is not a single protocol. It is a set of techniques for generating erase messages and distributing them to routers which may have cached offending content. We presented several alternatives, each of which are practical in different network locations. For instance, consumer-facing (caching) routers can keep lossless or lossy histories for at least a day. Meanwhile, interest marking is better suited for core network routers. Therefore, we believe that all aforementioned techniques can be used, in combination, for routing erase messages. Our specific recommendations are as follows:

1) If R supports interest marking, the first tuple in the hop-sequence traces is valid and appended by the router itself, then information in the tuple is used to route the erase downstream.
2) If the content is in R's cache, then in-cache history is used to route the erase.
3) If the content is not in R's cache, but R keeps lossless or lossy histories, then they are used for erase message routing.
4) Otherwise, R floods received erase messages.

[7]The average Internet hop-count is currently 16 [35].

[8]We defer designing a more efficient scheme for combining hop-sequence traces to future work.

Recommendation 1 is most appropriate for core network routers, 2 and 3 for less busy edge network routers, and 4 as a failover mechanism. Most routers would likely prefer to drop erase messages instead of flooding them. This is why BEAD is *best-effort*: it does not guarantee that each erase message will be delivered to all entities caching the target content.

As mentioned before, not all published content is subject to future deletion. If routers can make this distinction, there is no need to record history entries about content that will not be deleted. Such distinction can be achieved by adding an optional `CanERASE` flag to content object headers. If this flag is not present, the default behavior is to assume that no erase messages will ever be sent for the corresponding content. Moreover, interests requesting content that will not be deleted are not required to be marked by routers. Producers could tell consumers what content is subject to deletion (i.e., an erase) by overloading catalogs or manifests. As described in [7] and [36], catalogs and manifests contain lists of Self-Certifying Names (SCNs) of content to be requested. This list is provided by the producer and can contain the `CanERASE` flag alongside each SCN. In this case, the interest header format should be modified to include this optional flag. Moreover, since it is not guaranteed that all content objects will be requested using SCNs, the default behavior of (core) routers should be to append hop-sequence traces to interests if the `CanERASE` flag is missing.

VIII. Simulation Results

Our simulations focused on two properties of BEAD: network overhead (in terms of additional bytes added for erase messages) and forwarder overhead for processing erase messages, i.e., the average amount of time it takes to process each erase.

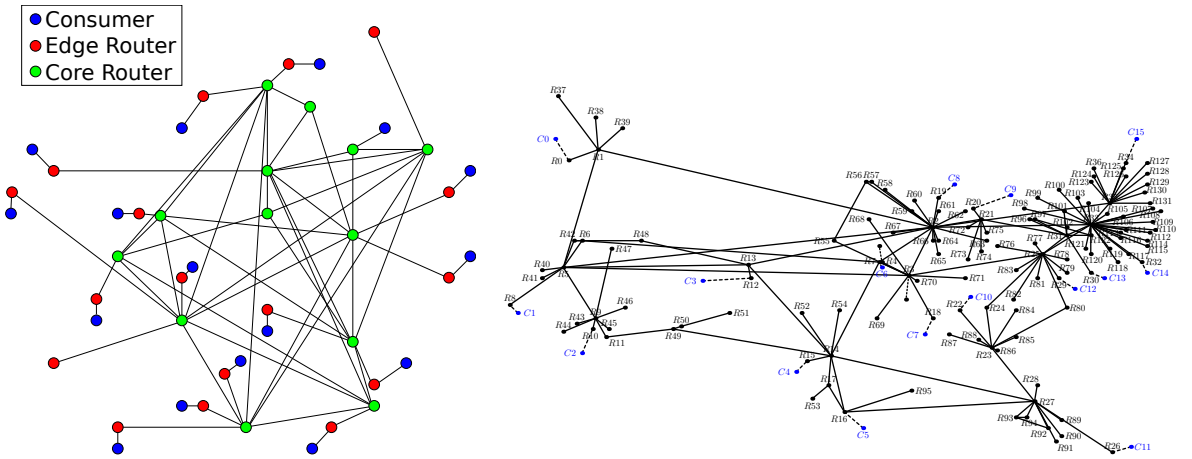
A. Network Overhead

To assess network overhead due to generating and forwarding erase messages we study the most costly scenario next to broadcasting: BEAD with lossless histories and routers with lossless links. To do so, we extended ndnSIM 2.0 [37] – an implementation of NDN architecture as a NS-3 [38] module for simulation purposes – to support erase messages. With this modified architecture, we ran two sets of experiments using the following topologies (shown in Figure 2):

- The DFN network, Deutsches ForschungsNetz (German Research Network) [39], [40]: a German network developed for research and education purposes which consists of 30 connected routers positioned in different areas of Germany. The blue dots in the figure represent group of consumers (10 consumers per blue dot) connected to edge routers (red dots), while the green dots represent core network routers.
- The AT&T backbone network [41]. This consists of over 130 routers. Each logical consumer in the figure represents multiple (5) physical consumers connected to an edge router.

In all experiments, consumers issue requests at a rate of 10 interests per second for content with the name prefix `/prefix/A` and monotonically increasing sequence number suffix. Every router uses a lossless history to record previously

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 185



(a) The DFN topology.

(b) The AT&T topology. Blue nodes represent clusters of consumers and black nodes represent routers.

Fig. 2. The DFN and AT&T topologies.

forwarded content objects for erase forwarding. Routers communicate over lossless links. Lastly, producers issue erase messages for 50% of their content every 1 second. (This may cause a producer to send a BEAD more than once.) Under these conditions, we measure router packet processing overhead with respect to content objects and erase messages. Figures 3(a) and 3(b) compare the overhead of processing content objects and erase messages in the DFN topology with 160 consumers. Similarly, Figures 3(c) and 3(d) show the same type of overhead in the AT&T topology with the same number of consumers. Comparatively, we find that erase messages contribute very little overhead to the network with respect to the bandwidth consumed by content objects. Specifically, the total amount of erase message traffic in the DFN topology is 1.8% of the total content objects traffic, whereas it is only 0.09% in the AT&T topology. To understand these differences, consider Figures 3(c) and 3(d). In Figure 3(c), core routers receive and forward more content packets than those not in the core. In Figure 3(d), those same core routers receive erase messages but do not forward all of them since they have were not in the history. This means that the content had previously been deleted. This is why the amount of egress traffic is less than the amount of ingress traffic.

We also assessed the actual computational overhead incurred by each router in these scenarios. The average time to process a single erase message for the DFN and AT&T scenarios are shown in Figures 4(a) and 4(b). We see that only a subset of the routers incur greater than 1.0ms to process an erase. These are the routers closest to the producer since they almost always receive, store, and forward erase messages.

IX. Monetizing Content Deletion

We now discuss potential economic incentives for routers and ISPs to support content deletion and implement BEAD.

A. BEAD & Accounting

So far, we discussed how the network routes erase messages towards routers that possibly cache corresponding content. The main challenge is that producers do not know where such content is cached. We also acknowledge that BEAD is best-effort, unless flooding is used, which is undesirable.

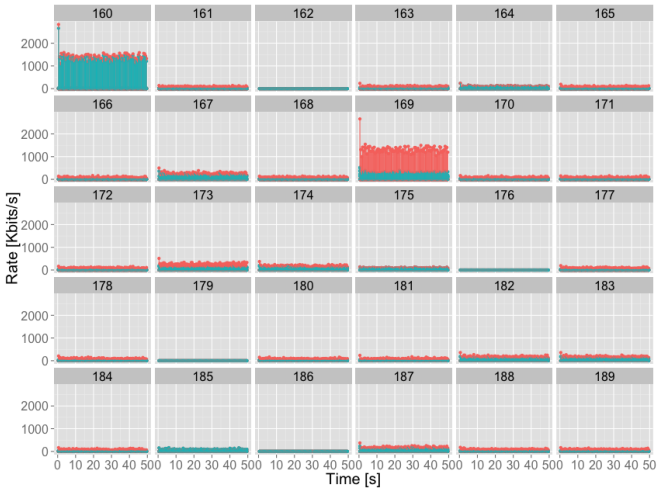
However, if producers knew exactly where content is cached, then erase messages could be routed efficiently. For example, if a producer knew that a particular AS had a copy of the content cached *by some node in the system*, then the producer could specifically ask the AS to distribute an erase internally. This is far superior to routing erase messages in the core of the network in hopes that they *might* reach this AS (and any others with a cached copy).

We believe that it is possible to distribute content caching location information along with accounting information. A scheme for secure accounting in CCN [42], suggests that routers should notify producers of content they serve from caches by sending a so-called "push interest" or *pInt*. This approach can be modified such that: (1) AS gateways send *pInt* messages when content is cached in their domain and (2) *pInt* messages carry the prefix of an AS accounting management server within the AS.[9] Whenever a producer wants to delete certain content, it sends an erase message to each accounting management server (one per AS) that previously reported caching corresponding content. Then, the latter distribute the erase message within their ASs. Intra-AS distribution can be achieved via techniques described in Section VI. In fact, flooding might well be appropriate for that purpose since erase messages would not traverse AS boundaries.

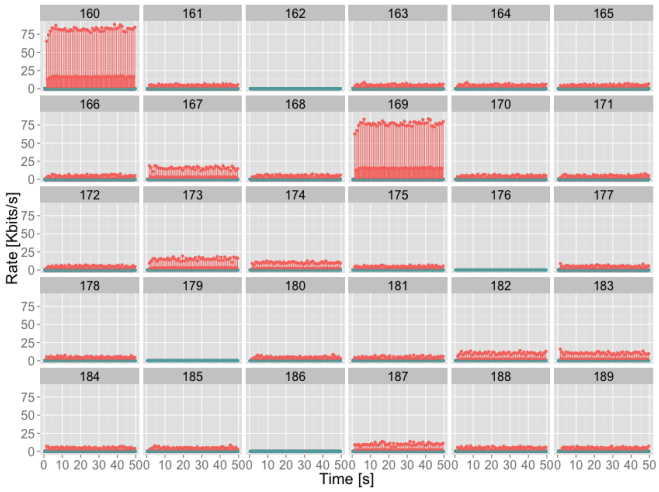
The relationship between accounting and BEAD is natural. This is because one of the important applications of accounting is to bill for cache space. From an economic perspective, it would not be surprising for in-network caching to become a paid service. Routers and ASs could offer caching services for producers. A reasonable extension to this service would be to also offer a deletion service via BEAD.

[9] Accounting management servers are centralized entities that manage accounting activities inside the AS.

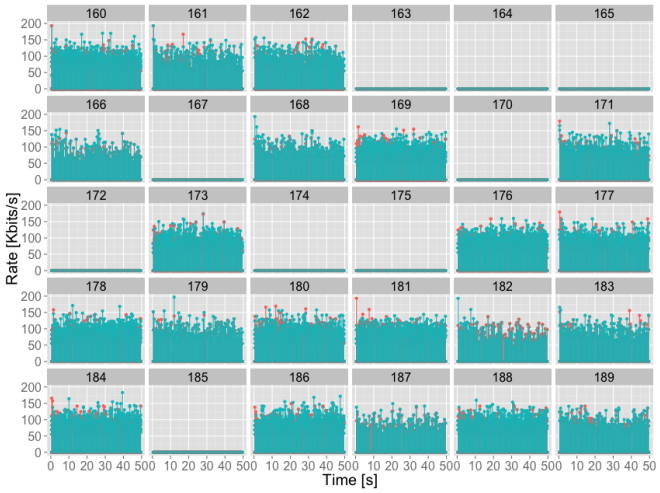
ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 186



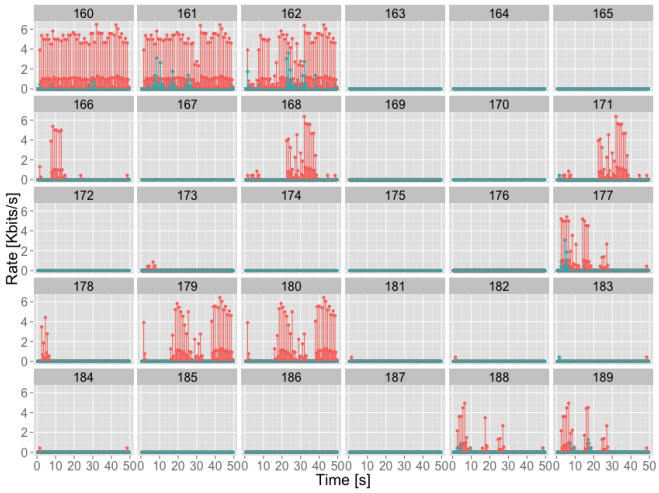
(a) Data processing overhead in the DFN topology with 160 consumers.



(b) erase message processing overhead in the DFN topology with 160 consumers.



(c) Data processing overhead in the AT&T topology with 160 consumers. Not all routers are present in the image.



(d) erase message processing overhead in the AT&T topology with 160 consumers. Not all routers are present in the image.

Fig. 3. Network overhead from processing erase messages. Routers are identified by integers in the range [160..189]. InData (OutData) and InErase (OutErase) correspond to the amount of content object and erase traffic received from (sent to) an upstream (downstream) node, respectively. Ingress data is shown in red and egress data is shown in blue.

B. BEAD in the Core

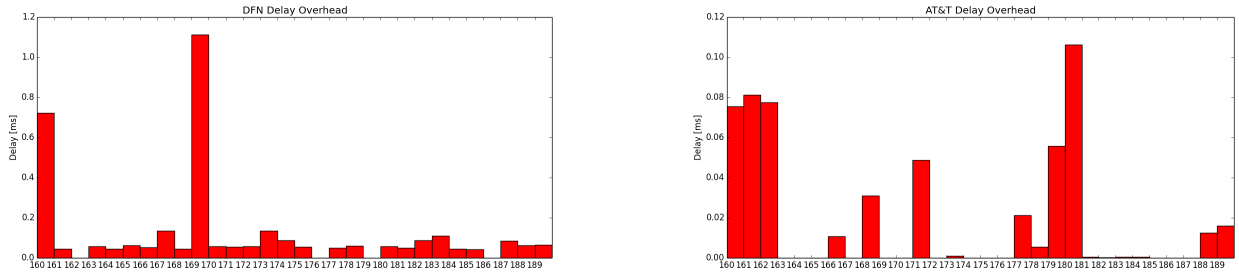
Flooding in the network core is not viable as a means of distributing erase messages. Moreover, forwarder histories and packet marking are (relatively) expensive operations and too costly for the fast path in the core. ISPs will likely just drop these messages due to a lack of economic incentive to forward them. Thus, in any plausible CCN network – where producers and consumers are at the edges of a network, while most traffic is routed through the core – erase messages are most likely to be propagated along only half of producer-to-consumer path(s). This is troublesome since content is most likely to be cached near consumers in edge (or near-edge) routers, and erase messages might never reach these routers.

To address this issue, core routers must be incentivized to carry and forward erase messages from producers to consumers. Since erase messages will typically amplify traffic, producers should be expected to pay for this increase. As before, this effectively turns BEAD into a service provided by ISPs that complements monetized caching; producers who pay for cache space may also have the choice to pay for on-demand deletion via BEAD.

X. Conclusion

We proposed BEAD – a technique for best-effort autonomous deletion in CCN. BEAD is designed to solve the problem of stale or unsafe content in CCN. We described an efficient and lightweight form of authenticator for BEAD deletion requests and discussed several ways in which they could be routed from producers to consumers. We assessed the performance of each technique and verified the network overhead using simulations. For future work, we will expand the set of experiments to study the penetration impact due to erase message forwarding based on lossy histories. We will

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 187



(a) DFN topology with 160 consumers.

(b) AT&T topology with 160 consumers.

Fig. 4. Forward erase processing overhead in the DFN and AT&T topologies. The results are captured for each of the routes assessed in the bandwidth overhead experiments. Routers are identified by integers in the range [160..189] and correspond to the routers in Figure 3.

also study this metric in the presence of lossy links. Finally, we will formalize the integration of accounting and BEAD to form a comprehensive platform for premium caching in CCN.

REFERENCES

[1] A. Dabirmoghaddam *et al.*, "Understanding optimal caching and opportunistic caching at the edge of information-centric networks," in *ICN*, 2014.

[2] M. Mosko and I. Solis, "CCNx semantics," 2015, https://www.ietf.org/id/draft-irtf-icnrg-ccnxsemantics-00.txt.

[3] A. Compagno *et al.*, "To NACK or not to NACK? negative acknowledgments in information-centric networking," in *ICCCN*, 2015.

[4] G. Mauri and G. Verticale, "Distributing key revocation status in named data networking," in *Advances in Communication Networking*, 2013.

[5] Y. Yu *et al.*, "An endorsement-based key management system for decentralized NDN chat application," *Technical Report NDN-0023*, 2014.

[6] C. Ghali *et al.*, "Interest-based access control for content centric networks," in *ICN*, 2015.

[7] C. Ghali *et al.*, "Network-layer trust in named-data networking," *ACM CCR*, vol. 44, no. 5, 2014.

[8] Y. Yu *et al.*, "Schematizing trust in named data networking," in *ICN*, 2015.

[9] C. Wood *et al.*, "Flexible end-to-end content security in ccn," in *CCNC*, 2014.

[10] F. Angius *et al.*, "Drop dead data," https://users.soe.ucsc.edu/~cedric/papers/angius2015drop.pdf.

[11] M. Myers *et al.*, "RFC 2560: X.509 internet public key infrastructure online certificate status protocol-ocsp," *Internet Engineering Task Force*, 1999.

[12] Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized dataset state synchronization in named data networking," in *ICNP)*, 2013.

[13] R. Ishiyama *et al.*, "On the effectiveness of diffusive content caching in content-centric networking," in *APSITT*, 2012.

[14] P. K. Agyapong and M. Sirbu, "Economic incentives in information-centric networking: implications for protocol design and public policy," *IEEE Communications Magazine*, vol. 50, no. 12, 2012.

[15] A. Araldo *et al.*, "Cost-aware caching: optimizing cache provisioning and object placement in ICN," in *GLOBECOM*, 2014.

[16] C. Wang and J. W. Byers, "Incentivizing efficient content placement in a global content oriented network," Technical Report BUCS-TR-2012-012, Boston University, Tech. Rep., 2012.

[17] A. Araldo *et al.*, "Cost-aware caching: Caching more (costly items) for less (ISPs operational expenditures)," *TPDS*, 2015.

[18] K. Suksomboon *et al.*, "On incentive-based inter-domain caching for content delivery in future internet architectures," in *AINTEC*, 2012.

[19] A. Araldo *et al.*, "Design and evaluation of cost-aware information centric routers," in *ICN*, 2014.

[20] A. C. Snoeren *et al.*, "Hash-based IP traceback," in *ACM CCR*, vol. 31, no. 4, 2001.

[21] M. T. Goodrich, "Efficient packet marking for large-scale IP traceback," in *CCS*, 2002.

[22] A. Belenky and N. Ansari, "IP traceback with deterministic packet marking," *IEEE communications letters*, vol. 7, no. 4, 2003.

[23] P. Gasti *et al.*, "Dos and ddos in named data networking," in *ICCCN*, 2013.

[24] F. Baker and P. Savola, "RFC 3704: Ingress filtering for multihomed networks," Tech. Rep., 2004.

[25] R. Stone *et al.*, "CenterTrack: An IP overlay network for tracking DoS floods." in *USENIX*, 2000.

[26] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, 1970.

[27] L. Fan *et al.*, "Summary cache: a scalable wide-area web cache sharing protocol," *TON*, vol. 8, no. 3, 2000.

[28] L. Zhang and Y. Guan, "Detecting click fraud in pay-per-click streams of online advertising networks," in *ICDCS*, 2008.

[29] G. Koloniari *et al.*, "One is enough: distributed filtering for duplicate elimination," in *CIKM*, 2011.

[30] F. Deng and D. Rafiei, "Approximately detecting duplicates for streaming data using stable bloom filters," in *SIGMOD/PODS*, 2006.

[31] H. Krawczyk *et al.*, "RFC 2104: HMAC: Keyed-hashing for message authentication," 1997.

[32] P. Gutmann, "RFC 6476: Using message authentication code (MAC) encryption in the cryptographic message syntax (CMS)," 2012.

[33] J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "Enabling correct interest forwarding and retransmissions in a content centric network," in *ANCS*, 2015.

[34] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, 2004.

[35] F. Begtasevic and P. Van Mieghem, "Measurements of the hopcount in internet," in *PAM*, 2001.

[36] J. Kurihara *et al.*, "An encryption-based access control framework for content-centric networking," in *IFIP Networking*, 2015.

[37] S. Mastorakis *et al.*, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," Technical Report, 2015.

[38] "Network simulator 3 (NS-3)," http://www.nsnam.org/.

[39] "DFN-Verein," http://www.dfn.de/.

[40] "DFN-Verein: DFN-NOC," http://www.dfn.de/dienstleistungen/dfninternet/noc/.

[41] A. Compagno *et al.*, "Poseidon: Mitigating interest flooding DDoS attacks in named data networking," in *LCN*, 2013.

[42] C. Ghali *et al.*, "Practical accounting in content-centric networking," in *NOMS*, 2016.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 188

Improving the Freshness of NDN Forwarding States

Jianxun Cao*, Dan Pei*, Zhelun Wu*, Xiaoping Zhang*, Beichuan Zhang†, Lan Wang‡, Youjian Zhao*
*Tsinghua University †University of Arizona ‡University of Memphis
*Tsinghua National Laboratory for Information Science and Technology (TNList)

***Abstract*—Named Data Networking (NDN) is a new Internet architecture that replaces today's focus on where – addresses and hosts – with what – the content that users and applications care about. A unique advantage of NDN over IP is the *adaptive* forwarding plane, which, by observing the performance of Interest/Data exchange, can dynamically select the best performing forwarding path, detect and recover from failures, load-balance across multiple paths, and mitigate attacks such as prefix hijacking and DDoS. A key component of adaptive forwarding is *interface ranking*, namely when and how to update the interfaces' metrics and rank them.**

As we will point out in this paper, however, the existing interface ranking scheme suffers from the problem of outdated forwarding states. Using two concrete problems, *SRTT slow-convergence* and *probing oscillation*, we illustrate how outdated forwarding states can impact the forwarding performance. We propose new forwarding strategies with *Adaptive SRTT Update (ASU)* and *Proactive Probing* to *achieve up-to-date forwarding states*, and evaluate how these strategies are able to address the two problems. Both theoretical analysis and simulation results show that the new strategies can reduce SRTT convergence time by 37.9% and the loss rate by 75% to 94.75%, compared to the existing interface ranking strategies.

I. Introduction

Named Data Networking (NDN) [1] is a new Internet architecture that emphasizes the content itself rather than its container (*e.g.*, host) or channel (*e.g.*, connection). In NDN, the consumer sends an Interest packet into the network to request a Data packet. The Interest carries the name of the data being requested instead of the destination address, and the matching Data packet can be retrieved from anywhere. By shifting the network service abstraction from "delivering a packet to the destination" to "retrieving a named data", NDN brings benefits such as in-network caching, native multicast, data-centric security and many others.

A unique feature of NDN is its *adaptive forwarding plane* [2], [3]. When an NDN router forwards an Interest packet out via a particular interface, it records this Interest in the Pending Interest Table (PIT) and starts waiting for a matching Data packet to return on the same interface. If the Data does return, the content retrieval is a success and the round-trip time (RTT) can be recorded to reflect the performance of using that interface. If the Data does not return in time or a Negative Acknowledgement (NACK) packet is received, the router knows that this interface does not work in retrieving this content and can record this information as well. This forms a forwarding-plane feedback loop that allows the router to detect any network fault, e.g., link failures and congestion, and take an alternative path to resolve the problem without relying on the control plane, i.e., routing convergence. Thus compared with IP's forwarding plane, NDN's is more intelligent, more resilient to network faults, and more effective in using multiple paths.

ISBN 978-3-901882-83-8 © 2016 IFIP

The decision process at the adaptive forwarding plane is called a "forwarding strategy," which considers the recorded performance metrics of multiple interfaces and chooses the most suitable one to forward an incoming Interest packet. A key component of forwarding strategies is the **interface ranking, *which updates the interfaces' metrics and ranks all the outgoing interfaces***, so that the strategy can choose the best interface to use. Without a good interface ranking scheme, forwarding states may be outdated, leading to reduced forwarding efficiency. In this paper, we focus on designing interface ranking schemes that can keep forwarding states up to date. We analyze two key parts of interface ranking: *periodical measurement* and *color classification*. The existing interface ranking schemes suffer from two problems: *SRTT slow-convergence* and *Probing oscillation*, which illustrates the impact of outdated forwarding states on forwarding performance. We propose new schemes for interface ranking to achieve up-to-date forwarding states, and show the effectiveness of our proposed solution via theoretical analysis and simulations.

The contributions of our work are twofolds. First, to the best of our knowledge, this paper is the first to study in depth interface ranking strategies and report two specific problems, SRTT slow-convergence and probing oscillation, which affect the accuracy of forwarding states. Second, we propose two new schemes, the *Adaptive SRTT Update (ASU)* algorithm and *Proactive Probing* approach, to achieve up-to-date forwarding states. Both theoretical analysis and NDNSim simulation results show that our new schemes reduce SRTT convergence time by 37.9% and the loss rate by 75% to 94.75%, compared to existing interface ranking strategies.

The remainder of the paper is organized as follows. Section II introduces basic concepts in NDN's adaptive forwarding, especially interface ranking. Limitation of current adaptive forwarding strategies is shown in Section III. SRTT slow-convergence and probing oscillation problems are described in Section IV and Section V, respectively. We introduce our new strategies and carry out the analysis theoretically in Section VI and Section VII, respectively. In Section VIII, we evaluate our new strategies using ndnSIM 2.0 simulator and analyze the results. Section IX briefly reviews related work. Finally, Section X presents our conclusion.

II. Adaptive Forwarding in NDN

In NDN, routers maintain state information of pending Interests, which brings adaptive forwarding plane to observe data retrieval performance and explore multiple forwarding paths. **Interface ranking** [3] is the key component of adaptive forwarding[1] to help routers find the current best outgoing interface to fetch data. *Color classification* and *periodical measurement* are introduced to help implement interface ranking.

Based on interface ranking, there are two major adaptive forwarding strategies in the literature[2]: Best-Route [3] and NCC [5]. These two approaches are similar in interface ranking and only differ slightly in the number of probed interfaces in color classification.

A. Name prefix

In NDN, Data names instead of IP addresses are hierarchically structured. For routing aggregation, router's FIB (Forwarding Information Base) contains *name prefixes*, and network routing protocols will distribute name prefixes in a way similar to distributing IP prefixes in todays Internet. A FIB entry records the working status of each interface. Please note that *interface status is per-name-prefix-per-interface*.

B. Periodical measurement and color classification

Detailed interface metrics (*e.g.* SRTT) are used for ranking interfaces. For a given prefix, to maintain the ranking metrics for interface ranking, the router periodically sends a copy of the Interest packet to all the interfaces to measure various metrics (such as SRTT) used in forwarding policies, and we name this probing as periodical measurement.

Besides ranking with interface metrics, color classification [3] is used to record an outgoing interface's working status for each prefix: *GREEN* means that the interface can bring the data back, *YELLOW* means that it is unknown whether the interface can bring the data back, and *RED* shows that the interface cannot bring the data back. The color classification are updated based on various feedbacks (whether to get the data successfully or not) from the network. Please note the periodical measurement can only change the ranking metrics, and it does not change the color of the interfaces regardless of the measurement results.

With periodical measurement and color classification, the interface ranking for a given prefix works as follows.

1) Ranking rules: For choosing the best path, GREEN interfaces are preferred over YELLOW ones. RED interfaces are not used in forwarding, and can change color only by the routing protocol. When ranking interfaces of the same color, NDN supports a wide variety of forwarding policies (or ranking metrics), such as "follow routing" based on OSPF cost, "the sooner the better" based on SRTT, and so on.

2) Forwarding based on ranking: For a specific Interest prefix, *all* the Interest packets destined to the same prefix will be transmitted through the best GREEN interface until the router receives a *NACK* or *Timeout*. The router will then degrade the currently used GREEN interface to YELLOW. If there still exists any GREEN interface, the router will switch to the highest ranked GREEN interface. Otherwise, the router will probe the YELLOW outgoing interfaces in the order of their ranking. We call the process of probing the YELLOW interfaces as **triggered probing**. We will explain the triggered probing in detail in the following.

C. Triggered Top-N probing

In Best-Route, once the probing process is triggered, the router will keep forwarding Interests along the current forwarding interface, which was just degraded from GREEN to YELLOW, and send a **probing** Interest to the highest ranked YELLOW interface to test whether it can bring Data back. When it receives NACK or Timeout for the probing packet, it will start trying the second-ranked YELLOW interfaces, and so on. On the other hand, if a Data comes back from the probed YELLOW interface, the router will change the color of this interface from YELLOW to GREEN, and this GREEN interface will be used as the best path, by now the triggered probing process is finished.

NCC is implemented in CCNx 0.7.2 as the default strategy. In NCC strategy, once the probing process is triggered, the router probes the top **2** YELLOW interfaces together if there are more than one YELLOW interfaces. Otherwise, it probes the only YELLOW interface. All other mechanisms are the same as in Best-Route. We unify these two approaches as **Triggered Top-N probing approaches**, where $N = 1$ in best-route, and $N = 2$ in NCC.

D. Benefits of interface ranking in adaptive forwarding plane

The periodical measurement and color classification with triggered probing in NDN's adaptive forwarding plane help a router maintain the information for all its interfaces. It allows the forwarding plane to quickly detect link fault through NACK mechanism and path probing. When a NACK arrives or any Interest packet is timed out, the router can freely explore the alternative path to find an available path to continue the forwarding work.

Compared with the routing plane, the forwarding plane can handle the link failure more quickly and solve the problem more flexibly. The forwarding plane can deal with the link failure without too much additional overhead as soon as possible when discovering any network fault.

[1]Adaptive forwarding consists of NACK, interface ranking, congestion control, and so on. In this paper, we just focus on interface ranking, and our study does NOT influence the strategy of NACK and congestion control.

[2]In this paper, we omit the impractical broadcast approach [1], which floods the Interest packet to all available interfaces when a router receives any incoming Interest packet. Obviously, for each Interest packet, broadcast approach can guarantee fetching the data from the optimal path, but it will tremendously add extra overhead. Furthermore, broadcast strategy breeds the Interest flooding attack [4] and causes trouble for network security in NDN. Thus, broadcast strategy has little practical application value except in some extreme cases.

III. Limitations of interface ranking

Interface ranking can help routers detect link fault more quickly, handle the link failure more promptly and solve the problem more flexibly. In ideal conditions, to choose outgoing interface, routers should make the decision to forward packets according to *the global network status*, because, for routers, the global network status can provide the optimal decision in real time to help routers optimize the outgoing packets flow. Specially, when the routers detect link fault, the global real-time network status can help routers ensure the essential network problem and figure out the most efficient solution. However, with the network measurement in the forwarding plane, the routers can just obtain *the local information*. Unlike the global information in routing plane, the local information in the forwarding plane limits the efficiency of forwarding packets and dealing with the link failure.

To approximate the global network status as far as possible, the network measurement for interface ranking metrics in forwarding plane and the ranking rules should be adaptive enough. However, the existing simple periodical measurement and color classification cannot work well enough.

A. Periodical measurement

With the in-network cache, the uncertainty of location where Interest packets are satisfied leads to the frequent changing of network metrics. Thus, in NDN, the simple periodical measurement will lead to some outdated network metrics. *In this paper, we focus on the important network metric for interface ranking: SRTT (Smoothed Round-Trip Time).* In Section IV, we will introduce a new problem caused by SRTT measurement: *SRTT slow-convergence*.

B. Triggered Top-N probing approach for color classfication

From Section II, we note that *Triggered Top-N probing approach is reactive*, which is illustrated in the following two key details. First, The probing is triggered after the NACK or Timeout, and YELLOW interfaces are probed in the order of their rankings. However, the ranking is based on periodical measurement, which can be quite outdated when there are **bursty congestions** along the path. Second, the change of interface color is done at the **packet-level** feedback for each prefix: NACK or Timeout for GREEN interfaces, and Data for YELLOW interfaces. When faced with burst congestions, the colors might frequently change between GREEN and YELLOW back and forth. In fact, later in Section V, we will show that, with the outdated ranking information under burst congestions, color oscillation will result from the packet-level action and will cause a new problem in NDN forwarding plane: *probing oscillation*.

IV. SRTT slow-convergence

In this section, we will focus on the measurement of SRTT and introduce a new problem: SRTT slow-convergence. SRTT reflects the network situation and is used for calculating RTO (Retransmission Time-Out) value. The reason why SRTT is used instead of RTT, is to eliminate as much as possible the impact of the network jitter which is caused by the instable network state such as the queue buffer congestion. In this paper, we adopt the typical algorithm: Exponential weighted moving average in TCP/IP to update the SRTT:

$$SRTT_i = \alpha \cdot SRTT_{i-1} + (1-\alpha) \cdot RTT_i \qquad (1)$$

Where α is the constant weighting factor ($0 < \alpha < 1$). Usually, α is between 0.8 and 0.9. In this paper, we let $\alpha = 0.8$ for the analysis and evaluation.

TABLE I
The FIB entry of R_c

prefix	interface				
	ID	OSPF	RTT	COST	COLOR
/prefix	R_2	0.55	0.7	1.27	GREEN
	R_3	0.67	0.73	1.4	YELLOW
	R_1	0.69	0.75	1.44	YELLOW
	R_4	0.75	0.8	1.55	YELLOW

TABLE II
Some notations

Con	The consumer
Pro	The producer
R_i	The routers
r	The loss rate of R_p
k	The number of sent Interest packets per second
t_{data}	The RTT of each outgoing interface
t_{out}	The timeout value

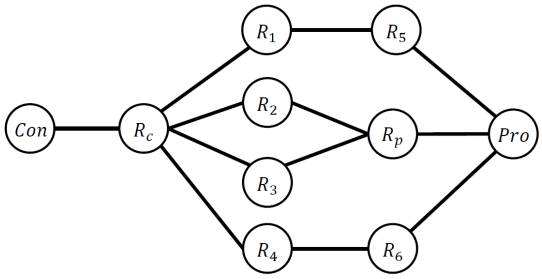


Fig. 1. A sub-topology of Tsinghua University Campus Network

Here is an example to describe SRTT slow-convergence. We use the sub-topology of Tsinghua University as shown in Fig 1 and give the COST[3] for interface ranking as shown in Table I. Some notations are defined in Table II.

We assume that at some time the consumer fetches the data along the path $Con - R_c - R_2 - R_p - Pro$. Here is a simple simulation to illustrate the problem. At the start, there is no cache in R_2 and R_p, so the consumer gets the data from the producer, which needs $150ms$. From $5s$ to $10s$, we assume that R_2 caches the needed data[4] and the consumer can just get the

[3]We specify COST to be $OSPF + RTT$ and the value of OSPF and RTT are both normalized linearly.

[4]For example, another consumer connected to R_2 fetches the same data from 5s to 10s.

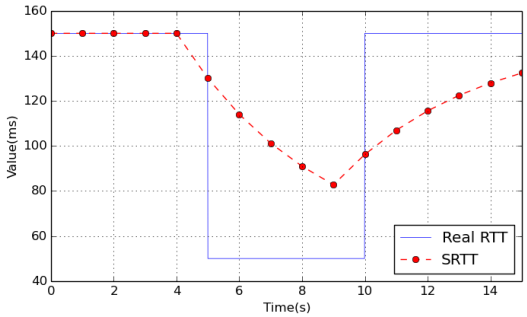


Fig. 2. A case to describe SRTT Slow-Convergence. In this case, the initial RTT is $150ms$. From $5s$, the RTT is reduced to $50ms$ and is recovered to $150ms$ after $10s$.

data from R_2 directly during this $5s$. We assume this RTT equals $50ms$. If the first consumer sends 100 Interest packets per second and the interval between two adjacent probing equals 100, then the RTT and SRTT perform as shown in Fig. 2. We can see that after $5s$, SRTT cannot reflect the real RTT. We call this problem as *SRTT Slow-Convergence*.

SRTT slow-convergence just exists in NDN while not in IP. In IP routing plane, the network metrics are nearly constant when the network works stably. Thus, SRTT filters the variation of RTT that are just caused by network jitter. However, in NDN, the variation of RTT cannot completely represent the network faults because of **in-network cache**. With the additional overhead, the frequency of periodical measurement cannot be very high, while the low measurement frequency will cause the SRTT value to not converge to the rapidly variational RTT timely, which makes SRTT often outdated. The outdated SRTT will deeply influence the judgement and lead to the above-mentioned SRTT Slow-Convergence.

V. Probing Oscillation

In this section, we introduce a new problem existing in NDN forwarding plane but not in IP: *Probing Oscillation*.

A. Problem description

Here is a simple example to illustrate what probing oscillation is. In Fig 1, the router R_c has four next hops R_1, R_2, R_3 and R_4. According to the interface ranking in FIB Table I, we see that firstly R_c will choose R_2 as the top choice to forward all the Interest packets.

Now we assume that $R_p - Pro$ congests, then, for R_c, the next actions of choosing the forwarding interface oscillate between the next two situations:

1) Situation 1: $R_2 \rightarrow R_3$: When congestion occurs, some Interest packets along $R_c - R_2 - R_p - Pro$ cannot fetch the corresponding data before the timer expires. But R_c just considers that R_2 now is not suitable to fetch data, so according to interface ranking, R_c will probe the second highest ranked interface R_3 to forward Interest packets along $R_c - R_3 - R_p - Pro$. It is worth noting that the link $R_p - Pro$ is the shared common sub-path of both paths ($R_c - R_2 - R_p - Pro$ and $R_c - R_3 - R_p - Pro$), thus for R_c, R_3 is still not the good choice. However, because congestion just causes packet loss instead of failure, even if the loss rate of $R_p - Pro$ reaches up to 10%, the success rate of probing is as high as 90%. Thus, *there is a very low possibility that a simple probing packet detects congestion, while for a given period of time, there is a very high possibility that congestion leads to timer expiration of some Interest packets.* So, if the router probes R_3 successfully, R_3 then becomes GREEN to forward Interest packets and R_2 changes to YELLOW.

2) Situation 2: $R_3 \rightarrow R_2$: Once the choice for R_c changes from R_2 to R_3, this choice is still instable and may change from R_3 to R_2. There are two possible reasons. First, the path $R_c - R_3 - Rp - Pro$ also congests (including the congested $pathRp - Pro$), then R_c will check the highest ranked YELLOW interface, namely R_2, and may switch back to R_2. Second, as illustrated in Fig. 3, we suppose at time t_0, R_c sends an Interest packet I_1 to R_2 which is lost later. So after t_{out} time, R_c does not receive the corresponding Data packet and mark the interface as YELLOW (Situation 1). But during this t_{out} tentative-sending time period, R_c has sent out $k \cdot t_{out}$ Interest packets to R_2 and some of these Interest packets, such as I_2, I_3, are satisfied after $t_0 + t_{out}$. This will cause R_2 to become GREEN again and R_2 replaces R_3 as the forwarding interface.

In conclusion, R_c will switch its outgoing interface back and forth between R_2 and R_3, and we name this phenomenon as *probing oscillation*.

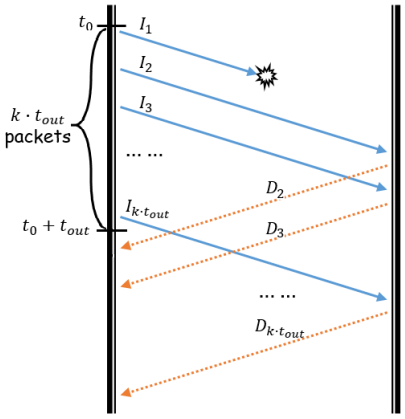


Fig. 3. A simple illustration of Probing Oscillation

The essential reason why probing oscillation occurs is the high-frequency color variation between GREEN and YELLOW caused by multipath forwarding with a shared congested common link and packet-level triggered probing, which makes that probing oscillation is unique in NDN while not in TCP/IP. In NDN, probing oscillation commonly happens and greatly influences the benefit of multipath forwarding according to our experiments. *When the router detects the network failure, with the influence of probing oscillation, it cannot recover from the network failure, which may bring a high loss of packets.* Although we just chose a simple topology to explain the problem, in a more large and complex network, probing

oscillation still continues to appear as long as there exists a shared common link, especially when there is a bottleneck link. Experiments in Section VIII prove the above claims.

VI. Adaptive SRTT Update (ASU) algorithm for periodical measurement

In this section and the next section, we propose some new strategies to improve the freshness of forwarding states to essentially solve the SRTT slow-convergence and probing oscillation. In this section, for periodical measurement, we firstly propose an **Adaptive SRTT Update (ASU) algorithm** to calculate SRTT more accurately with dynamic sample frequency instead of the simple periodical measurement. Then we will analyze the theoretical efficiency of solving the probing oscillation.

A. Adaptive SRTT Update (ASU) algorithm

Unlike IP, the forwarding plane in NDN can gain the dynamic network congestion status in real time, so that any router can update the cost of each outgoing interface per prefix according to the RTT of the pending Interest packets. Thus, all the interfaces need to update the SRTT value to reflect the network situation more exactly, which means routers need to flood a probing Interest packet periodically to all the interfaces. However, the probing frequency is hard to be set. On the one hand, low probing frequency will cause SRTT slow-convergence as described in Section IV. On the other hand, high probing frequency will cause too much extra overhead. In fact, in the most situation, SRTT tends to be constant except when the current RTT varies widely, *e.g*, cache hit. We hope to adopt *dynamical sample frequency* to make sure that the probing frequency can get lower when the RTT is approximately constant and get higher with the great variation of RTT.

Thus, we propose a new **Adaptive SRTT Update (ASU) algorithm** to calculate SRTT more accurately with dynamical sample frequency and improve SRTT slow-convergence. We use Δn to denote the number of Interest packets between two adjacent probing. The initial value of Δn is denoted as Δn_0 and the minimum or the maximum of Δn is denoted as Δn_{min} or Δn_{max}. We define the Changing Factor of RTT to be η which is calculated as follows:

$$\eta = \left| \frac{RTT - SRTT}{RTT + SRTT} \right| \tag{2}$$

From Equation 2, we can see that:

$$0 \leq \eta < 1 \tag{3}$$

Obviously, η is a normalized value to describe the distance between the real RTT and SRTT. The greater RTT changes, the bigger η will be. Based on the Changing Factor η, we propose the stretch Δn:

$$\Delta n_i = \begin{cases} \max(\Delta n_{min}, (1-\eta_i)\Delta n_{i-1}) & \eta \geq \eta_{threshold} \\ \min(\Delta n_{max}, (1+\beta)\Delta n_{i-1}) & \eta < \eta_{threshold} \end{cases} \tag{4}$$

In the above formula, $\eta_{threshold}$ denotes the critical value for distinguishing RTT changing greatly from RTT tending to SRTT. If η is larger than $\eta_{threshold}$, Δn should become smaller to increase the probing frequency. Otherwise, Δn should become bigger to decrease the probing frequency.

B. Analysis for solving SRTT slow-convergence

There are two cases when calculating SRTT: 1) When RTT changes little and 2) When RTT changes greatly. In case 1, no matter whether the ASU algorithm is used or not, SRTT will still be close to the real RTT, that is, all the strategies perform perfectly. However, by using the ASU algorithm, the Extra Overhead Ratio (EOR) will be lower:

$$EOR = \frac{1}{\Delta n_{max}} \tag{5}$$

In case 2, according to the definition of η in Equation 2, great variation of RTT means $\eta \geq \eta_{threshold}$. We assume that at 0 time, RTT changes from one value to another. Let $SRTT_0$ denote the initial SRTT at 0 time. We define m as:

$$m = \frac{RTT}{SRTT_0} \tag{6}$$

Obviously, when $\eta_i < \eta_{threshold}$, $SRTT_i$ approximately converges to RTT. From Equation 1, we can get the general term formula of SRTT:

$$SRTT_i = RTT \cdot \left[1 - \left(1 - \frac{1}{m}\alpha^i \right) \right] \tag{7}$$

Let $\eta_i < \eta_{threshold}$, then the probing number of quitting the SRTT-convergence i_c equals:

$$\begin{cases} i_c = \left\lceil \lg_\alpha \dfrac{2\eta_{threshold}}{(1-\eta_{threshold})(\frac{1}{m}-1)} \right\rceil, m < 1 \\ i_c = \left\lceil \lg_\alpha \dfrac{2\eta_{threshold}}{(1+\eta_{threshold})(1-\frac{1}{m})} \right\rceil, m > 1 \end{cases} \tag{8}$$

Assume that $m = 3$, $\eta_{threshold} = 0.1$, $\alpha = 0.8$, we can work out $i_c = 6$, which means, SRTT will converge to RTT after sampling for only 6 times. Thus, ASU algorithm is efficient for SRTT slow-convergence.

VII. Proactive Probing approach for color classification

In this section, for color classification, we introduce **Proactive Probing approach** as a new triggered probing approach, including introducing path backup and probability-based forwarding, to improve interface ranking and analyze the theoretical efficiency of solving probing oscillation.

A. Proactive Probing approach

The Proactive Probing approach contains two sides: 1) **constant Multi-GREEN-Interface strategy** to maintain several path backups, and 2) **probing based on *Probing Probability Function (PPF)* algorithm** for YELLOW interfaces to avoid probing strictly by interface ranking.

1) Constant Multi-GREEN-Interface strategy: The basic idea of constant Multi-GREEN-Interface strategy is that, for all the routers of which outgoing interfaces number is larger than 2, we set more than one GREEN interface to ensure some **backup paths**. The router will keep the number of GREEN interfaces as a constant **Minimal Number of GREEN Interfaces (MNGI)**. Constant Multi-GREEN-Interface strategy will start when an Interest packet comes to check the number of GREEN interfaces. While the number of GREEN interfaces is less than MNGI, the probing based on PPF mentioned in the following part will be triggered to choose one YELLOW interface to explore an available path. When this trying succeeds, the router will mark it as a GREEN interface. In order to prevent an extra cost, the value of MNGI should not be too large. We suggest that MNGI value does not exceed half the number of total interfaces.

2) Probing based on Probing Probability Function algorithm: As the above description, probing will be triggered when constant Multi-GREEN-Interface strategy discovers that the GREEN interfaces number is less than MNGI. However, how to probe will affect the performance directly. It is not smart enough that the router chooses the probing interface *only* based on the Interface Ranking. There are two reasons for this: 1) For each YELLOW outgoing interface, with the introduction of router cache, the OSPF value from the routing algorithm cannot reflect the actual time of fetching the data. 2) Because SRTT will not be updated since this interface was remarked as YELLOW, the SRTT similarly cannot reflect the actual efficiency of fetching the data, which means that the SRTT may be outdated. For example, in Fig. 1, the OSPF value of R_4 is the highest, but when the requested data is cached in R_4, choosing R_4 to fetch data is the optimal choice.

Thus, at this time, the COST attribute is not effective enough to rank all the YELLOW interfaces. But we can assume that *the smaller the COST is, the higher the probability of the optimal choice will be*. Based on this thought, we come up with the *Probing Probability Function (PPF)* to help **probability-based probing**. PPF should satisfy the decreasing monotonicity:

$$COST_i > COST_j \Leftrightarrow PPF(i) < PPF(j) \tag{9}$$

In this paper, we choose the following formula as PPF:

$$PPF(i) = \frac{1/COST_i}{\sum(1/COST_i)} \tag{10}$$

PPF ensures that each outgoing interface has a chance to be selected for probing. The probability is due to the history information of the COST calculated by both the routing algorithm and SRTT. Based on PPF, the router has a high probability of choosing the interface which performed well in the past, which guarantees the fairness for all interfaces.

B. Analysis of solving probing oscillation

The topology in Fig. 1 is selected for analysis. With Proactive Probing strategy, we choose the Best-Route (Triggered Top-1 probing) strategy for theoretical analysis. We try to analyze how many probing times the loss rate of R_c will converge to 0.

We assume that, at first, R_c chooses R_2 to fetch the data. When the first packet loss occurs, only if R_c doesn't choose R_3 as its next outgoing interface, then R_c jumps out of the probing oscillation. Thus, the probability of loss rate convergence equals $1 - PPF(3)$. Similarly, when the second packet loss occurs, the probability of quitting the probing oscillation equals $1 - PPF(2)$. Now we assume that the i^{th} packet loss occurs, then the probability of quitting the probing oscillation equals:

$$p = \begin{cases} 1 - PPF(2) & if \quad i\%2 = 1 \\ 1 - PPF(3) & if \quad i\%2 = 0 \end{cases} \tag{11}$$

We can approximatively consider it as Geometric distribution $G(p)$ while $p \simeq [(1 - PPF(2)) + (1 - PPF(3))]/2$. Thus, the expectation of the number of quitting probing oscillation (n_q) equals:

$$E(n_q) = \frac{1}{p} = \frac{2}{2 - PPF(2) - PPF(3)} \tag{12}$$

Even if we assume that $PPF(2)$ and $PPF(3)$ are as high as 0.8, $E(n_q)$ just equals 5. We can see that with Proactive Probing approach, limited probing times is enough to theoretically help routers jump out of probing oscillation.

VIII. Performance Evaluation

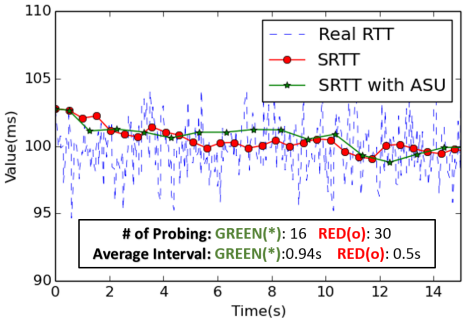
In this section, we evaluate the performance of both ASU algorithm and Proactive Probing approach for interface ranking in NDN using modified NDNSim 2.0 simulator [6]. The new version of NDNSIM integrates ndn-cxx library (NDN C++ library with eXperimental eXtensions) and the NDN Forwarding Daemon (NFD) to enable experiments with real code in a simulation environment. Our evaluation contains two parts. All the network parameters are set as follow:

- $t_{data} = 100ms$, $t_{out} = 120ms$
- $k = 100$
- $\eta_{threshold} = 0.1$

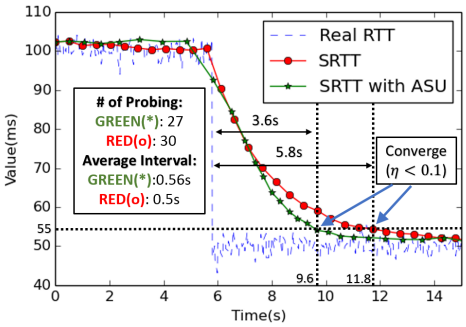
A. Evaluation on sub-topology of TUNET

We use the sub-topology of Tsinghua University Campus Network shown in Fig. 1 to verify the effectiveness of solving SRTT slow-convergence and probing oscillation.

1) Adaptive SRTT update algorithm for slow-convergence: Given the two different variations of generated Gaussian distributed RTT data, we use SRTT calculated by Equation 1 to see how the ASU algorithm has ameliorated the performance of RTT detection. **Firstly**: As shown in the Fig. 4(a), we set the distribution of RTT to be $N(100, 2)$ designed as a variable with very small variation. We see that no matter whether using ASU algorithm, SRTT can converge to real RTT as well. But paying attention to the overhead, we can find that in 15 seconds, with the advent of ASU algorithm, the total number of probing decreases from 30 to 16 and the average interval increases from 0.5s to 0.94s, which means that the Extra Overhead Ratio (EOR) is decreased by 46.7%.

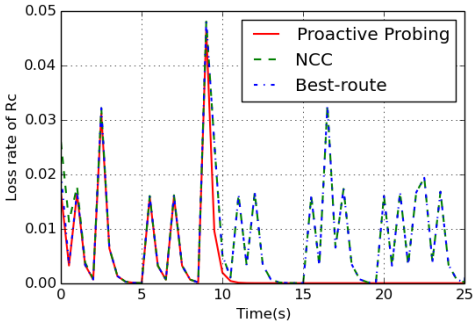


(a) SRTT comparison when RTT variation is small

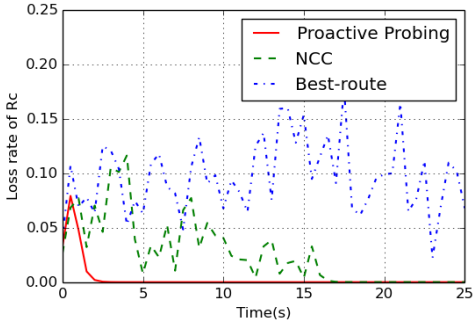


(b) SRTT comparison when RTT is suddenly reduced by half

Fig. 4. SRTT comparison when RTT variation is small or s suddenly reduced by half.



(a) The loss rate of R_c when $r = 0.01$



(b) The loss rate of R_c when $r = 0.1$

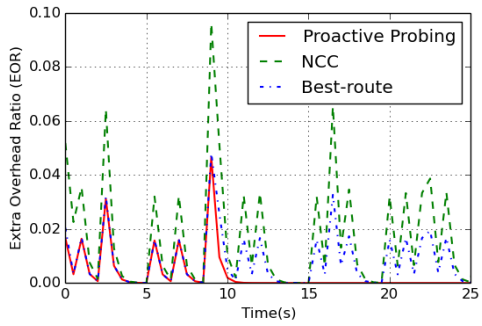
Fig. 5. The loss rate of R_c when $r = 0.01$ or $r = 0.1$. In a), the loss rate with NCC strategy and Best-route strategy nearly coincide while the loss rate with Proactive Probing strategy decreases to 0 at about 10 seconds. In b), the loss rate with Best-route strategy still performs the worst. The loss rate with NCC strategy converges to 0 after about 18s while the loss rate with Proactive Probing strategy decreases to 0 after only about 3 seconds.

So this proves that *in the flat phase of RTT, probing with ASU algorithm performs as well as the periodical measurement but brings a lower extra overhead.* **Secondly**: In this case, SRTT slow-convergence occurs. In the Fig. 4(b), we assume that there is a striking decline at around the time 6s. Before that, the distribution of RTT is $N(100, 2)$ and later is $N(50, 2)$. The figure illustrates that the SRTTs with ASU algorithm are more sensitive to the change. With ASU algorithm, the time from when RTT declines to it's convergence ($\eta < \eta_{threshold}$) is just 3.6 seconds, while on the other hand without ASU algorithm it's time is about 5.8 seconds. Thus, the response time is decreased by 37.9%. With ASU algorithm, although a little extra probing is generated during [6,10]s, the total overhead (average probing interval) in 15 seconds is also close to the plain probing. Thus, we conclude that *ASU algorithm can alleviate SRTT slow-convergence with little extra overhead as compared to the simple periodical measurement.*

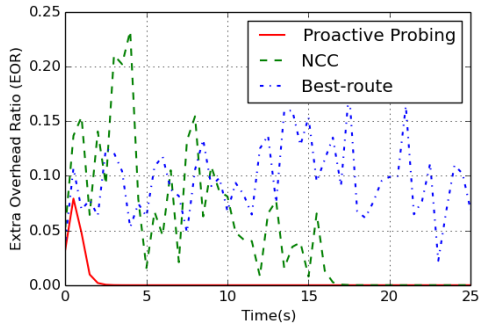
2) Proactive Probing approach for probing oscillation: To verify the performance of solving probing oscillation, we investigate two parameters: the loss rate of R_c and the extra overhead.

First, we analyze the loss rate of R_c in two occasions – one when the loss rate of R_p is 0.01 and the other is when the loss rate of R_p is 0.1. Both of the two cases will cause the probing oscillation of R_c. As shown in Fig. 5(a), when the loss rate of R_p is 0.01, we see that due to the very limited probability of packet loss, these three approaches perform closely to each other except that Proactive Probing approach slumps down to 0 at around 12s, while the other two keep on oscillating and do not reveal any trend to converge to 0. But as shown in Fig. 5(b), when the loss rate R_p is 0.1, things become more obvious. Proactive Probing approach leads to zero loss rate in only 3 seconds, which is about $1/5$ of the time as NCC falls to 0 loss rate. Best-route, however, does not show any trend of converging to 0. Therefore, we conclude from the experiments that Proactive Probing approach performs the best in solving the probing oscillation as compared to both the Best-Route and NCC.

Second, we see that the overhead of R_c is strongly related to the loss rate of R_c, because when the current path fails, each of the three strategies requires spontaneous probing of other pathes, and inevitably this incurs a detection overhead. Accordingly, from Fig. 6(a), we observe that when $R_p = 0.01$, the overhead of NCC is approximately 2 times as large as the loss rate of Best-route since NCC needs to find the first 2 available GREEN paths with the lowest cost. Best-Route and Proactive Probing approach, however, only need to find the cheapest GREEN path. And Proactive Probing approach performs as well as the Best-Route before it drops to 0 at



(a) Extra Overhead Ratio (EOR) when $r = 0.01$



(b) Extra Overhead Ratio (EOR) when $r = 0.1$

Fig. 6. Extra Overhead Ratio (EOR) when $r = 0.01$ or $r = 0.1$.

around the same time as the loss rate drops to 0. When $R_p = 0.1$, from Fig. 6(b) we still see that they almost copy the shape of the loss rate except for the 2-time-magnification of NCC.

B. Evaluation on a replica of the actual NDN testbed topology

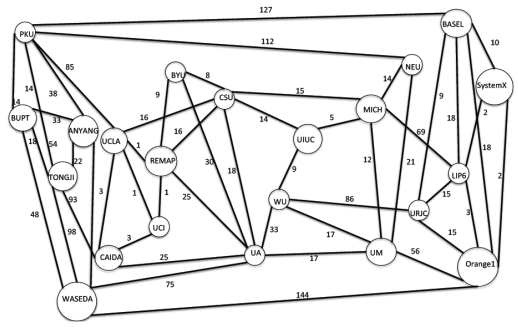


Fig. 7. A replica of the actual NDN testbed topology which contains 22 nodes and 50 links.

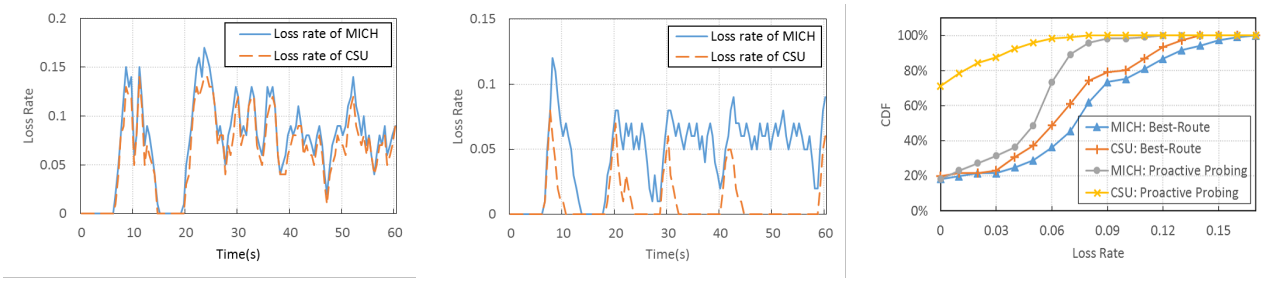
We use a replica of the actual NDN testbed topology which contains 22 nodes and 50 links shown in Fig. 7. The node "SystemX" is set as a server/producer and the other nodes are set as clients/consumers. The size of an Interest packet and a Data packet equals 25 Bytes and 1KB respectively. Each client initializes the sending rate of Interest packets as 10Kpps and randomly increases the sending rate by 5%-15% per second until the client detects the congestion which is defined in this section when the loss rate rises to 5%. Then the client will decrease the sending rate by half and repeat the increasing method. All the link bandwidth equals 1Gbps except for the bandwidth of MICH-LIP6 it is set as 100Mbps. *Because of the significantly lower bandwidth, the link MICH-LIP6 may become the bottleneck of this topology.*

We analyze the loss rate of the node MICH and the node CSU with the Best-Route and Proactive Probing approach respectively shown in Fig. 8. Because the link MICH-LIP6 is the bottleneck, it will get congested easily. Thus, it will cause one of its downstream nodes, CSU, to have probing oscillation. Fig. 8(a) shows that since the link MICH-LIP6 becomes congested, the node CSU begins to fall into probing oscillation and cannot handle this problem effectively with the Best-Route. Hence, the loss rate of CSU is almost as high as MICH's. While, in Fig. 8(b), with our Proactive Probing approach, although the congestion of MICH-LIP6 occurs, we can see that the loss rate of CSU decreases to 0 quickly. It means that CSU chooses another path to get data bypassing the link MICH-LIP6. We notice that after several seconds, the loss rate of CSU appears again and still returns to 0 in seconds. This is because that when the loss rate of MICH becomes low, CSU will reselect the path with MICH-LIP6 to get data until MICH-LIP6 seriously congests again. It's also worth noting that with Proactive Probing approach, the congestion level of MICH-LIP6 is significantly lower than that of Best-Route. With Proactive Probing approach, when CSU detects probing oscillation, it will finally unselect the congestion link MICH-LIP6, which will reduce the network flow via MICH-LIP6 and bring down the congestion level. Fig. 8(c) illustrates CDF of the loss rate with two approaches.

IX. Related Work

In TCP/IP, the stateless forwarding plane limits the network forwarding ability. Thus, multipath forwarding with which networks can provide end-hosts with multiple path choices are argued in [7] and [8] to perform better based on their experiments. Path Splicing [9], Pathlet routing [10] and Routing deflection [11] are designed to implement this idea. Multipath TCP [12] are introduced to set up multiple sub-TCP connections between the two ends. But all these improvement for TCP/IP forwarding plane is limited to the end-to-end transmission mode. Therefore, the increased adaptability of the forwarding plane are very limited. In TCP/IP, based on Fast reroute (FRR) mechanism [13], improved mechanisms such as MPLS FRR [14] and IPFRR [15] cannot efficiently handle multiple concurrent failures. For handling congestion control, Active Queue Management (AQM) mechanisms, such as Random Early Detection (RED) [16], have been introduced.

As an architecture of Information-Centric Networking (ICN), NDN supports multipath forwarding intrinsically. Adaptive forwarding plane [3] provide great performance in handling link failures and network congestion. Also, other congestion control mechanisms are studied. Hop-by-hop congestion control such as HR-ICP [17] and HIS [18] can take full advantage



(a) With Best-route strategy. (b) With Proactive Probing strategy. (c) The CDF of loss rate.

Fig. 8. The loss rate comparison between the node MICH and CSU in network testbed when using different strategies respectively.

of NDN adaptive forwarding plane, while all these mechanisms cannot effectively handle the dynamics in returning Data traffic. With AIMD algorithm liked in TCP/IP, Interest window size is presented to avoid excessive Interest packets [19], [20].

X. Conclusion

Adaptive forwarding plane is a key component for NDN architecture, but its existing strategies can suffer from outdated forwarding states, namely *interface coloring and ranking*. For the first time in the literature, in this paper we illustrate the impact of outdated forwarding states in the existing interface ranking mechanism through two concrete problems, *SRTT slow-convergence* and *probing oscillation*.

In order to efficiently achieve up-to-date forwarding states, the key challenge is to design *metric measurement* and *color classification* mechanism to balance the measurement overhead and the freshness of the forwarding states. Towards this goal, we propose *adaptive SRTT update* approach and *proactive probing* approach to improve the freshness of the forwarding states in NDN. Both theoretical analysis and NDNSim simulation results show that our new strategies reduce SRTT convergence time by 37.9% and the loss rate by 75% to 94.75% with little extra overhead, compared to the existing interface ranking strategies. We believe that this paper is an important first step towards up-to-date NDN adaptive forwarding plane.

Acknowledgement

This work has been supported by two National Natural Science Foundations of China (NSFC) under Grant No.61472210 and No.61472214, the Sate Key Program of National Science of China under Grant No.61233007, the National Key Basic Research Program of China (973 program) under Grant No.2013CB329105, the Tsinghua National Laboratory for Information Science and Technology Key Projects.

References

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc cllaffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 44, no. 3, July 2014.

[2] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking," *ICN(14): Proceedings of the 1st international conference on Information-centric networking*, 2014.

[3] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, April 2013.

[4] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," *IFIP Networking Conference*, May 2013.

[5] J. Shi, "ccnd 0.7.2 forwarding strategy," *http://redmine.named-data.net/projects/nfd/wiki/CcndStrategy, University of Arizona*, 2014.

[6] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim 2.0: A new version of the ndn simulator for ns-3," *NDN Technical Report*, vol. NDN, no. 0028, January 2015.

[7] D. Wendlandt, I. Avramopoulos, D. G. Andersen, and J. Rexford, "Dont secure routing protocols, secure data delivery," 2006.

[8] M. Caesar, M. Casado, T. Koponen, J. Rexford, and S. Shenker, "Dynamic route recomputation considered harmful," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 66–71, 2010.

[9] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path splicing," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 27–38.

[10] P. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 111–122, 2009.

[11] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 159–170.

[12] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp." in *NSDI*, vol. 11, 2011, pp. 8–8.

[13] P. Pan, G. Swallow, A. Atlas *et al.*, "Fast reroute extensions to rsvp-te for lsp tunnels," 2005.

[14] T. Nadeau, K. Koushik, and R. Cetin, "Multiprotocol label switching (mpls) traffic engineering management information base for fast reroute," RFC 6445, Tech. Rep., 2011.

[15] A. K. Atlas and A. Zinin, "Basic specification for ip fast-reroute: loop-free alternates," 2008.

[16] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 4, pp. 397–413, 1993.

[17] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 37–42.

[18] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee, "An improved hop-by-hop interest shaper for congestion control in named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 55–60, 2013.

[19] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin, "An empirical study of receiver-based aimd flow-control strategies for ccn," in *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*. IEEE, 2013, pp. 1–8.

[20] L. Saino, C. Cocora, and G. Pavlou, "Cctcp: A scalable receiver-driven congestion control protocol for content centric networking," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3775–3780.

Performance Evaluation of JT CoMP Approach: Tractable Model using Spatial Fluid Modeling

Lynda Zitoune[1,2], Stefan Cerovic[1], Danilo Cerovic[1], Véronique Vèque[1], Jean-Marc Kelif[3]
[1]Signals and Systems Laboratory, CentraleSupelec
3, Rue Joliot-Curie, 91192 Gif sur Yvette, France.
{lynda.zitoune,veronique.veque}@l2s.centralesupelec.fr
{stefan.cerovic,danilo.cerovic}@supelec.fr
[2]Department of Systems Engineering, ESIEE-Paris
2, boulevard Blaise Pascal, 93162 Noisy le Grand, France.
lynda.zitoune@esiee.fr
[3] Orange-Labs
38-40, rue du Général Leclerc, 92794 Issy Les Moulineaux, France.
jeanmarc.kelif@orange.com

***Résumé*—Inter-cell interference is a major issue in OFDMA networks, due to the increasing density of Low Power Nodes (LPN) used to offload the macro base stations. Coordination between these nodes also called Coordination MultiPoint (CoMP), is identified as a promising solution to improve the signal quality and the achievable throughput while ensuring spectral efficiency over the network. Joint Transmission (JT) mode of CoMP consists to jointly transmit the useful signal from more than one BS, typically the best serving one, and one or several other base stations. In dense networks, the performance evaluation and the analysis of JT CoMP approach become a hard task which needs lot of time and huge resources to conduct simulations. In this paper, we present a new mathematical framework based on spatial fluid modeling which reduces the analysis complexity and provides a macroscopic evaluation of the performance, quite faithful to those obtained using Monte Carlo simulations. The key idea is to consider a continuum of nodes rather than a fixed finite number, and to derive the mean impact of a density of nodes in a certain region of the network. The closed-form formulas of the downlink interference factor are defined for three scenarios depending on the number of coordinated nodes. Then, they are used to evaluate the signal quality improvement, particularly at the cell edge.**

***Keywords*—Interference mitigation, joint transmission, coordination multipoint, OFDMA network, spatial fluid modeling.**

I. Introduction

Today wireless networks become more and more dense in terms of the number of base stations (BSs), access points (APs) and mobile systems (MSs). Increasing the density of BSs and APs is one of solutions to face the exponential growth of wireless data traffic, to improve the critical application QoS (Quality of Service), and to fulfill the increasing demand of users [1]–[4].

Dense cellular networks are of two types [1]–[3]. In urban areas, operators often resort to additional cellular infrastructure to fulfill the users demand and to ensure application QoS, making the already existing cellular network more dense [5]. Furthermore, heterogenous networks are also considered as dense where macro BSs are used together with several Low Power Nodes (LPN) such as, pico or femto BSs, to form a k-tier network [6], [7]. Small cells are used to offload macro BSs and hence enhance the coverage and the capacity of the overall network and improve the throughput of the end users [8]. However, due to the limited radio resources, the interference is unavoidable and leads to neutralize the benefits of **densification** by decreasing the quality of the signal received by the MSs mainly at the cell edge [9]–[12], and consequently reducing the network throughput.

Many solutions are used to resolve the interference impact between adjacent cells, also called inter-cell interference [9], [13]–[15]. For example ICIC technique (Inter-Cell Interference Coordination) (release 8 of LTE) and its enhanced version (e-ICIC) (release 10) are based on some resource scheduling, frequency reuse or power control to reduce the interference at the cell edge in homogeneous and heterogeneous networks respectively. Nevertheless, these approaches present some limitations in case of saturation, i.e. when the terminal number grows, and induce a serious issue on throughput reduction. Recently, Coordinated MultiPoint approach (CoMP), under standardization by 3GPP for LTE-Advanced technology (release 11), is considered as a promising solution to provide high spectral efficiency where destructive interference is turned to constructive one. Some projects have dealt with the practical CoMP schemes in both downlink and uplink communications, and assessed their performance using simulations and field trials to demonstrate the maturity of such approaches as in [9], [13]. Joint Transmission (JT) and Coordinated Beamforming/Scheduling (CB/CS) are two ways to mitigate interference using coordination [9], [12], [16]. In the former, the coordination is performed based on data sharing/exchanging between coordinated BSs. In the last one, coordination is based on the channel state information exchange in order to select the appropriate beams avoiding interference.

In this paper, we focus on the Joint Transmission method of CoMP when applied in OFDMA network where no intra-cell interference exists. Radio resources in OFDMA are parallel and orthogonal. Technically, a User Equipment (UE) receives multiple signals of the same information from several eNodeBs (denoted eNBs in this paper) together with its serving eNB.

ISBN 978-3-901882-83-8

All these eNBs form a coordinated set. As a consequence, JT coordination improves the cell throughput since all resource blocks are used to transmit only useful information. It enhances the signal quality, mainly at the cell edge, and extends the BS coverage [17]. A new control plan is defined to integrate the coordination, also called CloudRAN, to design the coordinated set, and to deal with issues related to synchronization and resource scheduling on the backhaul network between involved eNBs. Some results related to these issues can be found in [9], [13], [18].

To model the operation of JT CoMP, we develop a fluid modeling framework to evaluate its performance on the downlink, in terms of interference factor decrease and its impact on the signal quality received at the UE. The key idea we consider here, is that a continuum number of eNBs, rather than a fixed finite one, is spatially distributed in the network. We derive a tractable model of the downlink interference factor [19] when JT CoMP is used in three different scenarios. We use the underlying model to compute the mean of interference for different UEs placed randomly in the network and also to evaluate the SINR, the main gain metric of JT CoMP. This spatial fluid model, provides a coarse-grained characterization of the network, by considering eNBs density rather than an exact distance that separates each eNB from each UE. Unlike works based on stochastic geometry modeling, which give the average of the performance (like SINR, outage/coverage and throughput) in a typical position, at a given cell [20]. Moreover, stochastic geometry models are most often not tractable, when others point processes rather than PPP (Poisson Point Process) are used to describe the node positions.

The paper contributions are as follow :

- We provide a mathematical model of JT CoMP using a spatial fluid framework, which is tractable making the evaluation of the signal quality easier using simple expression, whatever the location of a UE.
- We investigate the gain of the JT CoMP approach using the obtained model, by considering different number of eNBs in the coordinated set and environment parameters.
- We prove the effectiveness of the spatial modeling as a mathematical framework for dealing with the interference and as a performance evaluation tool even when the eNBs density decreases due to the coordination. We compare the obtained results to those of Monte Carlo simulations of an equivalent hexagonal grid model.

The resulting model of JT CoMP is a powerful tool to investigate the impact of the coordination on the whole network, and not only on a given cell, since the coordination induces more than one cell and enables UEs in neighboring cells to benefit from coordination. So, the model can be easily used by the network operator as a dimensioning/planning tool for coordination between the BSs. We point out that the computational complexity of the model is out of the paper scope, as well as the comparison to the spatial Poisson process. In the latter case, some results can be found in [21].

The paper is organized as follows. In section II we present some related work on JT CoMP modeling using mainly spatial point processes. Next, we introduce a background on the fluid modeling paradigm and the interference factor in section III. Afterwards, the system model of JT CoMP is explained and analytical expressions are detailed for three scenarios depending on the size of the coordinated set in section IV. In section V, we present the numerical results of the underlying analytical expressions and discuss the accuracy of the model toward Monte Carlo simulations of an equivalent hexagonal one. Finally, conclusions and some perspectives are presented in section VI.

II. Related work

CoMP is a coordination technique which involves multiple nodes or BSs to reduce interference at the cell edge and hence increases the network throughput and the spectral efficiency of the radio channel. The coordination is performed between BSs by exchanging data in Joint Transmission (JT) or channel state information in Coordination Beamforming/Scheduling (CB/CS) [9], [14]. In JT, a UE receives the same data from multiple eNBs in the coordinated set defined beforehand. The eNBs in the coordinated set use the same radio resources and as a consequence improve the received signal quality. Commonly, the coordinated set is formed by neighboring eNBs closer to the UE or which provide strongest signal. Moreover, the number of coordinated eNBs is limited in order to address the technical and practical challenges of the backhaul network [9], [13].

Several works dealt recently with the interference issue in both homogeneous and heterogeneous cellular networks, using different modeling approaches and methodologies, for example : game theory [22], simulation and trials [9], [13], and stochastic geometry [5]–[7]. A concise overview on stochastic geometry can be found in [20], [23]. To be inline with the contributions of our work, we present some works on the modeling and the performance evaluation of CoMP using mainly stochastic geometry. In [18], the authors characterize the SINR distribution when coordination multipoint is used and discuss some practical design problems. The main result of this work is that the SINR is increasingly improved, when the number of BSs increases in a ball with a fixed radius. Therefore the gain of cooperation, in terms of coverage, increases with the path-loss exponent. An evaluation of the coverage probability of an heterogeneous network described by a Poisson Point Process is presented in [24]. Two different connectivity models are considered for coordination, 1) n-strongest BS connectivity model where the coordinated set is composed of BSs which provide strongest signal. 2) n-nearest BS connectivity model, where the coordinated set is composed of BSs close in each tier. The analysis shows that the n-strongest model is better than the n-nearest one. In [25], the authors consider CB/CS method of CoMP and show that the performance metrics decrease linearly in case of a non ideal backhaul with a large delay. In [26], it is demonstrated that JT is more powerful than CB/CS in terms of performance. The improvements in the network performance are approved in [17] using a realistic urban scenario. Depending on the difference between the received signal strength of the serving cell and the coordinated cells an interference map is constructed. The CoMP gain is then derived using the interference map.

A common remark is that all these studies are carried out considering a Poisson Point Process [11], [20], [23], substantiated by the fact that the nodes location is often random and leads to irregular networks. Tractable models are consequently provided for the SINR distribution, coverage/outage ratio and

the average rate over the network, where different propagation models (fading/shadowing) are considered. However, when other Point Processes are assumed, mainly regular one [27]–[29], the performance models like SINR and its derivatives are not analytically tractable due to the non-independent nature of points. In this case, either approximations are used to bound the performance parameters [20], or intensive simulations are conducted to validate the models [30], [31].

Motivated by the spatial fluid modeling developed in [19] and [32], we propose here to use this mathematical framework in order to investigate the benefits of JT CoMP in a dense cellular network, and at the same time, the accuracy of the yielding solutions. To meet this objective, we consider a downlink channel of OFDMA network and derive explicit, numerically tractable integral expressions of the interference factor for three different scenarios of coordination. The interference factor in this case depends on several parameters : position to the serving base station, density of BSs in the vicinity and the path-loss function which is distance-dependent. We compute SINR, the key metric, more precisely the SIR as we neglect the noise effect.

III. Background materials

Let us consider a single frequency OFDMA wireless network composed of B base stations (BS) (denoted eNodeB (eNB) in this paper), which covers a urban area. We focus on the downlink. The radio resources of a base station are divided in a number of parallel, orthogonal, non-interfering channels (subcarriers), each one transmitting at power P. Therefore, only inter-cell interference is considered. The User Equipments are randomly distributed over the network.

A. Interference Factor

As defined in [19], the interference factor at the user equipment u is defined as the ratio of total power received from other base stations $p_{ext,u}$, to the power it receives from its serving BS b, $p_{int,u}$, such as : $f_u = \frac{p_{ext,u}}{p_{int,u}}$. Since in OFDMA the sub-carriers are assumed to be orthogonal, therefore intra interference does not exist, so $p_{int,u} = P_b g_{b,u}$ is the useful power, P_b is the power transmitted by the eNB b, and $g_{b,u}$ is the inverse of the path-loss between the serving eNB and the UE u. The interference factor can be expressed as : $f_u = \frac{1}{P_b g_{b,u}} \sum_{j \neq b}^{B} P_j g_{j,u}$, where P_j denotes the power transmitted by eNB j, and $g_{j,u}$ the inverse of the path-loss between the eNB j and the UE u. B represents the number of eNBs considered in the cellular radio system.

As a consequence, the quality of the received signal characterized by the Signal to Interference plus Noise Ratio defined as :

$$SINR = \frac{P_{b,u}}{P_{ext,u} + N}$$

At the UE u, the SINR can be expressed as :

$$SINR = \frac{1}{f_u + \sigma} \tag{1}$$

Where N is the Gaussian noise and $\sigma = \frac{N}{P_{b,u}}$. Let notice that since all subscribers transmit at the same power P, we can write for each subscriber :

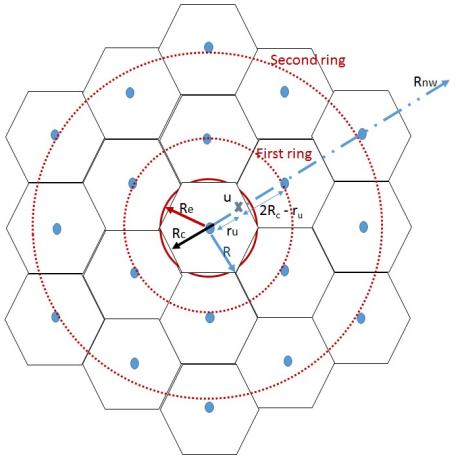


FIGURE 1. The network fluid model and the equivalent hexagonal model.

$$f_u = \frac{1}{g_{b,u}} \sum_{j \neq b}^{B} g_{j,u} \tag{2}$$

In the case of an homogeneous network, f_u depends only on the number of base stations, their positions, the positions of UEs, and their corresponding path-losses [19].

B. Short overview of fluid model

In fluid network modeling [19], [32], the main assumption is that a fixed finite number of eNBs is replaced by an equivalent continuum of eNBs which are spatially distributed in the network. So basically, the transmitting power of all the eNBs is considered as a continuum field all over the network, and in the case of a uniform eNB distribution and a uniform traffic, the network is considered to be homogeneous. Considering a path-loss model given by [33], $g_{b,u}$ is expressed as $Kr^{-\eta}$ where K is a constant, r represents the distance between a UE u and its serving BS, and $\eta > 2$ is the path-loss exponent. To reduce the complexity of the model, we neglect the effect of the shadowing and focus our analysis on the other parameters (the shadowing will be considered in a future paper). Therefore, f_u also depends on distance r. So f_u in the equation (2) can now be written as a function of r and not anymore as a function of u. A new parameter ρ_{eNB} is introduced which represents the density of BSs. It is constant since the network is assumed to be homogeneous, so that all the eNBs j have the same output power P_j. As in [19], we will consider a circle shaped network around the cell of interest, where half of the distance to the nearest base station is R_c, as shown on the Figure 1. We will assume that the whole network radius is R_{nw}, and that there are no transmitters at the distance greater than that.

The following distances of interest will be used throughout this paper, and they are represented on the Figure 1 : 1) R the hexagonal cell radius, 2) R_c half of the distance to the nearest eNB, 3) R_e the radius of an equivalent disc (i.e. a disc with the same area as the hexagon).

C. Network without coordination : fluid model

As the assumption of a continuum BS is considered in the fluid paradigm, to calculate the amount of external power at certain location in a cell, we can sum up the influence of each small subsurface, $zdzd\theta$, in the area of interest in the network. In other words, the external power received at the UE u located at the distance r_u from its serving eNB, can be calculated by the integration of $\rho_{eNB} z dz d\theta P_b K z^{-\eta}$ over the ring with inner radius of $2R_c - r_u$, and the outer radius of $R_{nw} - r_u$, as shown on the Figure 1 (see [19] for more details).

$$\begin{aligned} p_{ext,u} &= \int_0^{2\pi} \int_{2R_c - r_u}^{R_{nw} - r_u} \rho_{eNB} P_b K z^{-\eta} z dz d\theta \\ &= \frac{2\pi \rho_{eNB} P_b K}{\eta - 2} [(2R_c - r_u)^{2-\eta} - (R_{nw} - r_u)^{2-\eta}] \end{aligned} \quad (3)$$

The useful power received by the UE u can be calculated as

$$p_{b,u} = P_b K r_u^{-\eta} \quad (4)$$

because in this case the only serving base station is b. So, combining the last two equations, we can calculate the interference factor as :

$$f_u = \frac{2\pi \rho_{eNB} r_u^{\eta}}{\eta - 2} [(2R_c - r_u)^{2-\eta} - (R_{nw} - r_u)^{2-\eta}] \quad (5)$$

We notice that the interference factor depends only on the distance r from base station b, so it can be written as a function of r, like :

$$f(r) = \frac{2\pi \rho_{eNB} r^{\eta}}{\eta - 2} [(2R_c - r)^{2-\eta} - (R_{nw} - r)^{2-\eta}] \quad (6)$$

Note that in the last expression we assumed that $\rho_{eNB} = (3\sqrt{3}R^2/2)^{-1}$, so that, it is constant in every point of the network, because of the assumption that traffic is uniform. We can conclude that the interference factor does not depend on the output power of the eNB (homogeneous network), but only on the density of eNBs ρ_{eNB}, the radius of a cell R_c, the size of the considered network R_{nw}, and the path-loss parameter η, which means it can be easily calculated.

We consider coordination between the serving BS b and eNBs of the first ring, since they are the potential nodes which entail the signal quality received by UEs.

IV. System Model of a network with JT-CoMP

In this part, we propose the application of the fluid model when the cooperation between base stations exists in the network, i.e. when JT CoMP technique is used. The issues related to data sharing and synchronization on the backhaul are not considered here. Let $f_u^{'}$ denote the interference factor when JT CoMP is used, so that :

$$f_u^{'} = \frac{P_{ext}^{'}}{P_{int}^{'}} \quad (7)$$

The impact of JT CoMP on the calculation of the interference factor is given through the reduction of the external power p_{ext} given in (3), by the amount of power P_{CoMP} that UE receives from the eNBs which are in cooperation with its serving eNB b. This power amount P_{CoMP} is added to the internal power given in (4).

$$f_u^{'} = \frac{P_{ext} - P_{CoMP}}{P_b + P_{CoMP}} \quad (8)$$

We define a new factor $G_{CoMP} = \frac{P_{CoMP}}{P_{int}}$ as the gain of JT CoMP using the fluid model. The interference factor $f_u^{'}$ in this case, depends on the f_u without coordination and can be defined as follows :

$$f_u^{'} = \frac{f_u}{1 + G_{CoMP}} - \frac{G_{CoMP}}{1 + G_{CoMP}} \quad (9)$$

As in fluid model there are no exact borders between different cells, in order to express the impact of the coordinated eNBs on the UE u, we have to define the area of the cooperation, mainly the integration domain and to sum up the influence of each small subsurface, $zdzd\theta$ over it. We define the boundaries of the first ring as $2R_c - r_u$ and $4R_c - r_u$ as depicted in Figure 1 for the polar axis, since closest eNBs are at the distance $2R_c$ from serving eNB.
The polar angle depends on the number of eNBs in the coordination set. Since in hexagonal lattice model there are six eNBs neighbors around the serving eNB, we propose to divide the first ring into six equal areas, and suppose that each area has the same impact on the UE because of the assumption of homogeneous network. Therefore, we define the integration boundaries of the polar angle from 0 to $n\frac{\pi}{3}$ (because $2\pi/6 = \pi/3$), n is the number of coordinated eNBs considered in the first ring. As given in the following formula, the power of coordinated stations $b_{1,\cdots,n}$ is expressed as :

$$p_{b_{1,\cdots,n},u} = \int_0^{\frac{n\pi}{3}} \int_{2R_c - r_u}^{4R_c - r_u} \rho_{eNB} P_b K z^{-\eta} z dz d\theta$$

and is equal to :

$$p_{b_{1,\cdots,n},u} = \frac{n\pi}{3} \frac{\rho_{eNB} P_b K}{\eta - 2} [(2R_c - r_u)^{2-\eta} - (4R_c - r_u)^{2-\eta}] \quad (10)$$

Three different scenarios are considered in the following to evaluate the JT CoMP gain when one, two, three eNBs are in the coordinated set.

A. Scenario 1 : cooperation with one base station in the first ring

As the simplest case, cooperation between the serving eNB b and one base station in the first ring, b_1, is considered. In order to express the impact of the eNB b_1 on the UE, we must do the integration over the one sixth of the first ring, because there are six base stations in the first ring, assuming that each one of them has the same impact on the UE (Figure 2).

As given in the following formula, the power of the coordinated station b_1 is :

$$p_{b_1,u} = \frac{\pi}{3} \frac{\rho_{eNB} P_b K}{\eta - 2} [(2R_c - r_u)^{2-\eta} - (4R_c - r_u)^{2-\eta}] \quad (11)$$

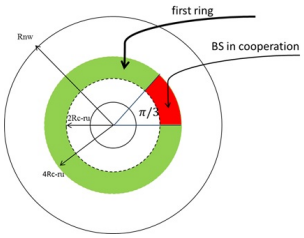


FIGURE 2. Scenario 1 : coordination with one eNB from the 1^{st} ring

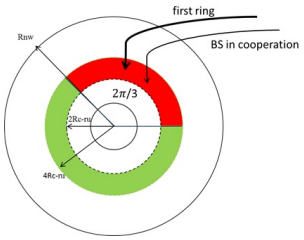


FIGURE 3. Scenario 2 : coordination with two eNBs from the 1^{st} ring

So, in this scenario, the total external power is :

$$p_{e1}=2\pi\frac{\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(R_{nw}-r_u)^{2-\eta}] - \frac{\pi}{3}\frac{\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(4R_c-r_u)^{2-\eta}] \quad (12)$$

The total internal power is :

$$p_{i1} = P_bKr_u^{-\eta}+\frac{\pi}{3}\frac{\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(4R_c-r_u)^{2-\eta}] \quad (13)$$

We can calculate the interference factor by dividing p_{e1} defined in (12) by p_{i1} of equation (13). The improvement compared to the default case in terms of interference factor is obvious since we increase the nominator and decrease the denominator with the same fraction.

B. Scenario 2 : Cooperation with two eNBs in the first ring

In this case cooperation between serving eNB b and two eNBs ($n = 2$), b_1 and b_2, in the first ring is considered,as in Figure 3. So, the reasoning is the same like in the previous case, with the only difference that now two base stations out of six are included in the calculation of the useful power :

$$p_{b_{1,2},u} = \int_0^{2\pi/3}\int_{2R_c-r_u}^{4R_c-r_u}\rho_{eNB}P_bKz^{-\eta}zdzd\theta \quad (14)$$

This will give us the following expressions for the external power p_{e2} and the internal power p_{i2}, respectively :

$$p_{e2}=\frac{2\pi\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(R_{nw}-r_u)^{2-\eta}] - \frac{2\pi}{3}\frac{\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(4R_c-r_u)^{2-\eta}] \quad (15)$$

$$p_{i2} = P_bKr_u^{-\eta}+\frac{2\pi}{3}\frac{\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(4R_c-r_u)^{2-\eta}] \quad (16)$$

In this case the interference factor, which we can get by dividing (15) by (16), is even higher than in previous case.

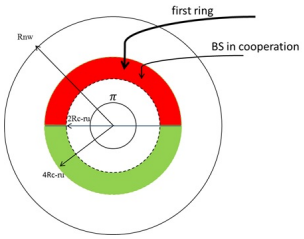


FIGURE 4. Scenario 3 : coordination with three eNBs from 1^{st} ring

C. Scenario 3 : cooperation with 3 BSs in the first ring

In this scenario, we consider the cooperation between serving the eNB b and three eNBs in the first ring (b_1, b_2 and b_3) as shown in Figure 4. Now, the external interference is reduced and the internal one is increased by the following amount :

$$p_{b_{1,2,3},u} = \int_0^{\pi}\int_{2R_c-r_u}^{4R_c-r_u}\rho_{eNB}P_bKz^{-\eta}zdzd\theta \quad (17)$$

because three base stations out of six make an angle of exactly π. The external and the internal interference are given with following formulas in this case :

$$p_{e3}=\frac{2\pi\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(R_{nw}-r_u)^{2-\eta}] - \pi\frac{\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(4R_c-r_u)^{2-\eta}] \quad (18)$$

$$p_{i3} = P_bKr_u^{-\eta}+\pi\frac{\rho_{eNB}P_bK}{\eta-2}[(2R_c-r_u)^{2-\eta}-(4R_c-r_u)^{2-\eta}] \quad (19)$$

As expected, the interference factor grows even more compared to previous cases.

V. SIMULATION AND NUMERICAL RESULTS

The objective of this section is twofold. First, we aim to validate the model of JT CoMP for the three scenarios exposed earlier. We compare numerical results of the fluid interference factor to those obtained by Monte Carlo simulations of an equivalent hexagonal model. The second objective is to evaluate the gain of the coordination by computing the SINR and comparing its variation to the case where no coordination is applied between eNBs.

For Monte Carlo simulation, we consider 10 rings of hexagonal cells around a central hexagon of interest such that $R_{nw} = 21R_c$. 50 UEs are generated uniformly in the central

Parameters	Values
R	50 m (femtocell network)
R_c	43.30 m ($R_c = \frac{R\sqrt{3}}{2}$)
η	$\in \{2.5, 3, 3.5, 4\}$
p_t	250 mW
Bandwidth	10 MHz

TABLE I. SIMULATION PARAMETERS.

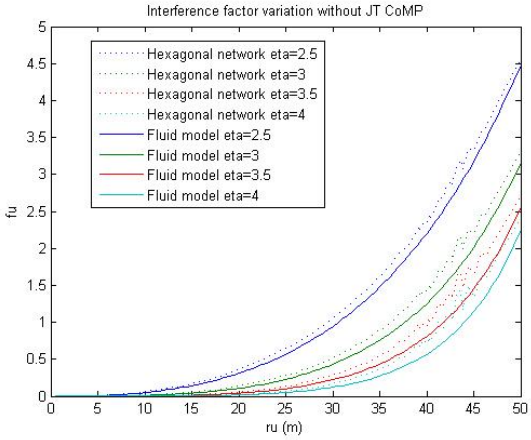


FIGURE 5. Interference factor variation without JT CoMP

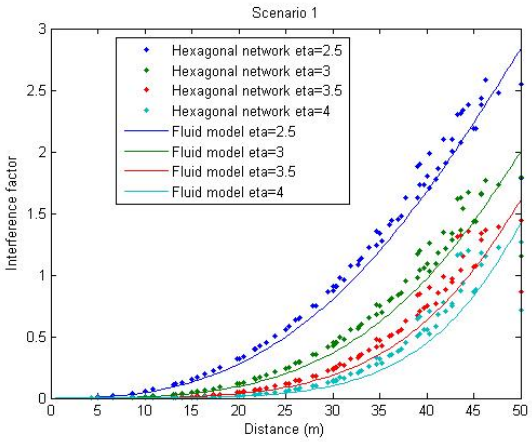


FIGURE 6. Interference factor variation in scenario 1

	no CoMP	Scenario 1	Scenario 2	Scenario 3
$\eta = 2.5$	4.48	2.84	2.40	1.4
$\eta = 3$	3.16	2.00	1.60	0.92
$\eta = 3.5$	2.55	1.61	1.23	0.70
$\eta = 4$	2.25	1.42	1.04	0.60

TABLE II. NUMERICAL RESULTS OF f_u AT THE CELL EDGE, r_u=50M.

hexagon and we suppose that they are attached to the eNB located at the center of this hexagon. We sort all these UEs depending on the distance to their serving eNB, and we average the interference factor for all the UEs at the same distance. To simulate the coordination, the coordinated eNBs are chosen uniformly out of 6 possible eNBs from the first ring depending on each scenario.

The numerical results of fluid modeling are obtained using equations, (6) (without coordination), (12) and (13) for scenario 1, (15) and (16) for scenario 2 and (18) and (19) for scenario 3, over a network of radius R_{nw}. The eNBs density is $\rho_{eNB} = [\frac{3\sqrt{3}}{2}.R^2]^{-1}$ in a central circle of radius R (cf Figure 1, R is drawn in blue arrow). The other simulation parameters common with hexagonal model, are summarized in the table I.

The plot of the interference factor as a function of distance from the serving eNB for the default case, without coordination, is shown in Figure 5. The curves of this figure prove the accuracy of the fluid model towards hexagonal lattice one. These results are pretty similar to those in [19] and [32], and show an obvious result that is, near the serving base station, the interference factor is lower than at the edge. For example for $\eta = 2.5$, the interference factor is about 0.25 at 20m from the serving eNB, and reaches 2.1 at 20m further. f_u increases exponentially with the distance whatever the exponent values, and it is inversely proportional to the path-loss exponent, i.e. f_u is higher for a lower loss path exponent $\eta = 2.5$.
Furthermore, in the default case (Figure 5), we can notice that there is a slight difference between fluid and hexagonal model. This difference is bounded and does not exceed 8%. The difference is related to the circular symmetry around the serving eNB, and the circular shaped form considered in fluid model. So, whatever the position of the UE in the inner circle, the average of the external power of all neighboring eNBs is the same. However in the hexagonal model, this assumption is no longer valid.

When eNBs of the first ring are used together with the serving one, the interference factor decreases significantly as shown in Figures 6, 10 and 12. In the table II, we give the fluid numerical results of f_u at 50m, at the cell edge for different path-loss exponent. For example for $\eta = 2.5$, we observe that the interference factor decreases by a factor of 3.1 when three eNBs are added to the coordinated set.

The plots of SIR (Signal to Interference Ratio) in dB, in Figures 7, 11 and 13 show that fluid model of JT CoMP match very well with Monte Carlo simulations of Hexagonal model. Moreover, the gain of coordination is highlighted in Figures 8 and 9. The gain is increasingly important at the cell edge, and

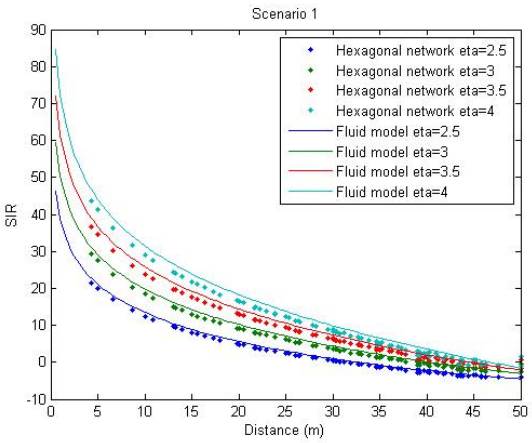


FIGURE 7. SIR variation in case of scenario1

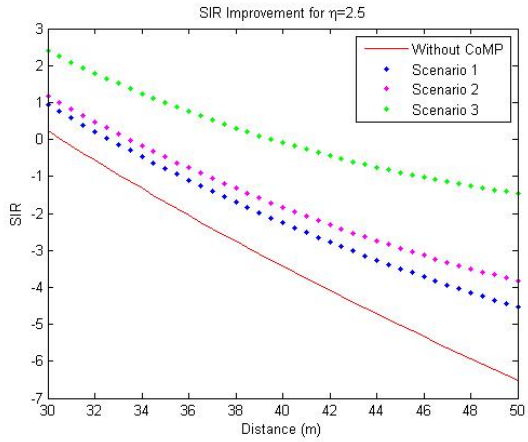


FIGURE 8. SIR improvement, $\eta = 2.5$

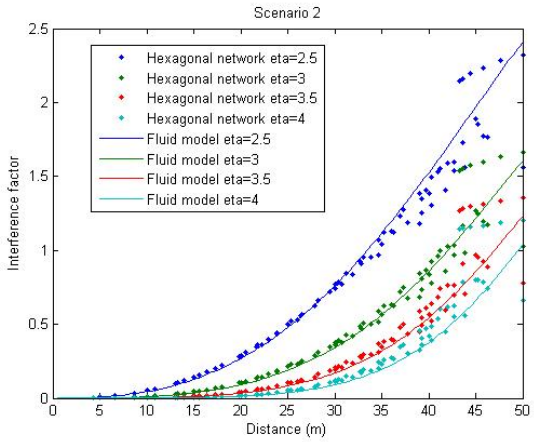


FIGURE 10. Interference factor variation : scenario2

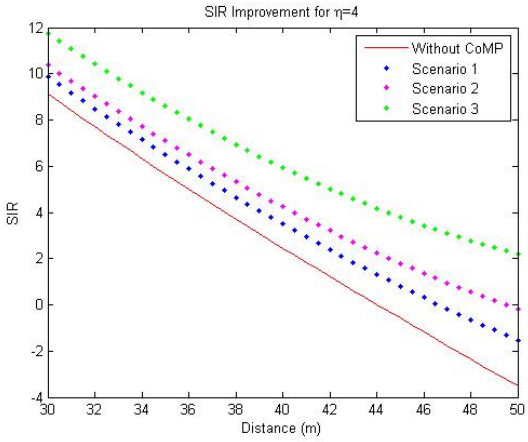


FIGURE 9. SIR improvement, $\eta = 4$

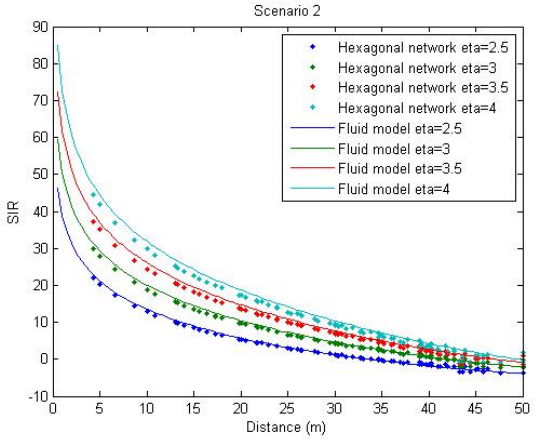


FIGURE 11. SIR variation in case of scenario2

we notice that it is quite related to the coordinated BSs number and to the distance from the serving eNB than the path-loss exponent η. For example at the distance of 46 m from the serving eNB, the SIR gain is about 4.2 dB for $\eta = 2.5$, and about 4.5 dB for $\eta = 4$. This gain is observed considering the default scenario without CoMP and the scenario3. At the same distance, comparing curves of scenario without CoMP and scenario 2, we observe that the CoMP gain is around 2 dB, in Figures 8 and 9, whatever the value of η. More generally, the gain of cooperation is two times more important every time a new eNB joins the coordination process, whatever η.

In the second scenario, 2 eNBs from the first ring cooperate with the serving eNB. They are picked uniformly out of 6 possible eNBs from the first ring around the serving eNB. The plot of the interference factor for this scenario in Figure 10, shows that there is a difference between fluid and hexagonal models, mainly for UEs which are further away from the serving eNB. The difference here is quite higher than that observed for the first scenario in the Figure 6 which is around 12%.

In the third scenario, 3 BSs from the first ring cooperate with the serving BS. They are also picked uniformly out of 6 possible BSs from the first ring around serving BS, as can be seen in the Figure 4. The plot of the interference factor for this scenario is shown in Figure 12. We notice that there is also a distinction between fluid and hexagonal model for the UEs which are further away from the serving BS. In both scenarios 2 and 3, the difference between the fluid and hexagonal models is quite higher than the scenario 1, which is around 12%.

In case of cooperation with one, two or three eNBs of the first ring, the impact of these BSs is subtracted from the external power and added to internal power as shown in the previous section. In the fluid model, the impact of the cooperation is not related to only the eNB or eNBs involved in the cooperation, and the distance separating this or these BSs to the UE. The impact of coordination in fluid modeling is related to a density of eNBs over the sub-area around the UE, which is constant. Moreover, fluid modeling inherently assumes a circular symmetry around the UE in case of coordination. Consequently, whatever the UE position, removing the impact of one, two or three sub-areas from the first ring remains the same. However, in the hexagonal model exact distances between coordinated eNBs and the UE are considered. Furthermore, since coordinated eNBs are picked uniformly out of 6 possible BSs, the coordination gain of far BSs is insignificant compared to the remaining external power. Another point which can explain the difference between the

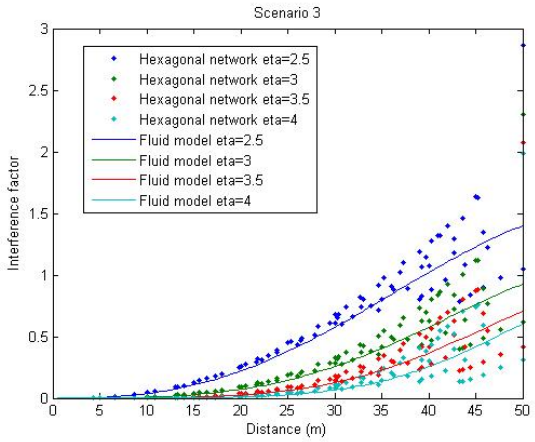


FIGURE 12. Interference factor variation : scenario3

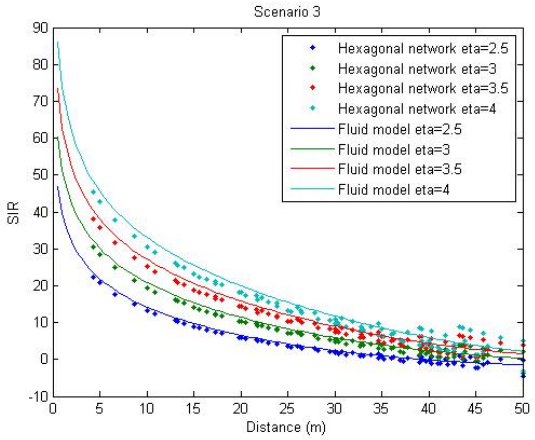


FIGURE 13. SIR variation in case of scenario3

fluid an the hexagonal models, is related to the difference between the area of the hexagon of radius R and the equivalent circle of radius Re. So that, the fluid model does not catch the extreme parts of the hexagon.

Indeed, the fluid model gives analytical expressions which do not take into account probability that user will be positioned differently from the CoMP eNB. This is a very interesting feature, since the UE is not compelled to perform measurements to define the best coordinated eNBs, and thus simplifies the multipoint association procedure.

VI. CONCLUSION

The Joint Transmission of Coordinated MultiPoint approach is used to deal with the inter-cell interference, the main issue emphasized by the increased number of BSs and LPNs (Low Power Nodes). In this paper, we developed a new mathematical framework based on spatial fluid modeling to investigate the gain of this approach on the downlink of an OFDMA network. We considered a continuum number of eNBs spatially distributed in the network rather than a fixed finite number. We derived closed-form formulas of the interference factor for three scenarios depending on the number of coordinated eNBs. These tractable expressions allow us to rapidly compute the SINR of UEs in an OFDMA network.

Numerical results show that the gain of JT coordination is related to the coordinated BSs number and to the distance from the serving eNB than to the path-loss exponent η. The gain is two times more important every time a new eNB joins the coordination process. Furthermore, the proposed framework reduces considerably the analysis complexity and provides a macroscopic evaluation of the performance, faithful to those obtained using Monte Carlo simulations. Indeed, fluid modeling inherently assumes a circular symmetry around the UE in case of coordination, and gives analytical expressions which do not take into account probability that user will be positioned differently from CoMP eNB. The last result is very interesting, since the UE is not compelled to perform measurements to define the best coordinated eNBs, and thus simplifies the multipoint association procedure.
Future work will focus on the characterization of this difference between the fluid and the hexagonal models mainly in the cell edge, by finding an expression for an upper bound of the difference. We aim also to develop expressions of the other metrics like the throughput and the coverage probability, and to investigate the effectiveness of the fluid framework considering the shadowing effect.

ACKNOWLEDGEMENT

This work is part of the French project LCI4D of the Systematic Cluster, Paris Region Systems & ICT Cluster (http://www.systematic-paris-region.org/).

RÉFÉRENCES

[1] C.-X. Wang and al, "Cellular architecture and key technologies for 5G wireless communication networks," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 122–130, Feb. 2014.

[2] S. Yuas, M. Valkama, and J. Niemela, "Spectral and energy efficiency of ultra-dense networks under different deployement strategies," *IEEE Communications Magazine*, vol. 53, no. 1, Jan. 2015.

[3] B. Soret, K. Perdersen, N. Jorgensen, and V. Fernandez-Lopez, "Interference coordination for dense wireless networks," *IEEE Communications Magazine*, vol. 53, no. 1, Jan. 2015.

[4] M. Bennis, M. Simsek, A. Czylwik, W. Saad, S. Valentin, and M. Debbah, "When cellular meets WiFi in wireless small cell networks," *Communications Magazine, IEEE*, vol. 51, no. 6, 2013.

[5] J. Andrews, F. Baccelli, and R. Ganti, "A tractable approach to coverage and rate in cellular networks," *IEEE Transactions on Communications*, vol. 59, no. 11, pp. 3122–3134, Oct. 2011.

[6] H. Dhillon, R. Ganti, and J. Andrews, "A tractable framework for coverage and outage in heterogeneous cellular networks," in *Information Theory and Applications Workshop (ITA)*, Feb. 2011, pp. 1–6.

[7] H. Wang, X. Zhou, and M. Reed, "Analytical evaluation of coverage-oriented femtocell network deployment," in *International Conference on Communications (ICC)*, June 2013, pp. 5974–5979.

[8] V. Chandrasekhar, J. Andrews, and A. Gatherer, "Femtocell networks : A survey," *CoRR*, vol. abs/0803.0952, 2008.

[9] R. Irmer, H. Droste, P. Marsch, S. Brueck, H.-P. Mayer, L. Thiele, and V. Jungnickel, "Coordinated multipoint : Concepts, performance, and field trial results," *IEEE Communications Magazine*, Feb. 2011, iMT-Advanced and Next-Generation Mobile Networks.

[10] A. Daeinabi, K. Sandrasegaran, and X. Zhu, "Survey of intercell interference mitigation techniques in LTE downlink networks," in *Australasian Telecommunication Networks and Applications Conference (ATNAC)*. IEEE, Nov. 2012, pp. 1–6.

[11] H. Dhillon, R. Ganti, F. Baccelli, and J. Andrews, "Modeling and analysis of K-Tier downlink heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 3, pp. 550–560, April 2012.

[12] D. Lee, B. Clerckx, E. Hardouin, D. Mazzarese, S. Nagata, and K. Sayana, "Coordinated multipoint transmission and reception in LTE-Advanced : Deployment scenarios and operational challenges," *IEEE Communications Magazine*, pp. 148–155, Feb. 2012.

[13] V. Jungnickel and al, "Field trials using coordinated multi-point transmission in the downlink," 21st International Symposium on Personal, Indoor and Mobile Radio Communications Workshops, 2010.

[14] S. Singh, A. Kumar, S. Khurmi, and S. Tanvir, "Coordinated multipoint (CoMP) reception and transmission for LTE-Advanced/4G," *International Journal of Computer Science And Technology*, vol. 3, June 2012.

[15] A. S. Hamza, S. S. Khalifa, H. S. Hamza, and K. Elsayed, "A survey on inter-cell interference coordination techniques in OFDMA-based cellular networks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1642–1670, 2013.

[16] W. Yu, T. Kwon, and C. Shin, "Multicell coordination via joint scheduling, beamforming and power spectrum adaptation," in *IEEE INFOCOM 2011*, April 2011, pp. 2570–2578.

[17] S. Berger, Z. Lu, R. Irmer, and G. Fettweis, "Modelling the impact of downlink CoMP in a realistic scenario," in *IEEE Wireless Communications and Networking Conference (WCNC)*, April 2013, pp. 3932–3936.

[18] R. Tanbourgi, S. Singh, J. Andrews, and F. Jondral, "A tractable model for non-coherent joint-transmission base station cooperation," pp. 4959–4973, Sept. 2014.

[19] J.-M. Kelif and E. Altman, "Downlink fluid model of CDMA networks," in *Proc. of IEEE VTC Spring*, May 2005.

[20] R. Heath and M. Kountouris, "Modeling heterogeneous network interference using poisson point processes," in *Information Theory and Applications Workshop (ITA)*, Feb. 2012, pp. 17–22.

[21] J.-M. Kélif, S. Senecal, C. Bridon, and M. Coupechoux, "Quality of service and performance evaluation : A fluid approach for poisson wireless networks," in *IEEE International Conference on Network of the Future (NoF)*, Paris, France, Dec. 2014.

[22] K. Khawam, A. A. S. Lahoud, J. Cohen, and S. Tohme, "Game theoretic framework for power control in intercell interference coordination," in *IFIP Networking Conference*, June 2014, pp. 1–8.

[23] B. Blaszczyszyn and H. Keeler, "Studying the SINR process of the typical user in poisson networks by using its factorial moment measures," *CoRR*, vol. abs/1401.4005, 2014.

[24] G. Nigam, P. Minero, and M. Haenggi, "Coordinated multipoint in heterogeneous networks : A stochastic geometry approach," in *IEEE GLOBECOM Workshop on Emerging Technologies for LTE-Advanced and Beyond 4G*, Atlanta, GA, Dec. 2013.

[25] P. Xia, C.-H. Liu, and J. Andrews, "Downlink coordinated multi-point with overhead modeling in heterogeneous cellular networks," *CoRR*, vol. abs/1210.5503, 2012.

[26] Y. Yifan, R. Yun, L. MingQi, S. Bin, and S. RongFang, "Achievable rates of coordinated multi-point transmission schemes under imperfect CSI," *In Proc. IEEE ICC*, pp. 1–6, 2011.

[27] M. Haenggi, "Mean interference in hard-core wireless networks," *IEEE Communications Letters*, vol. 15, no. 8, pp. 792–794, Aug. 2011.

[28] M. Abdelrahman, T. ElBatt, and A. El-Keyi, "Coverage probability analysis for wireless networks using repulsive point processes," in *Personal Indoor and Mobile Radio Communications (PIMRC)*, Sept 2013, pp. 1002–1007.

[29] I. Nakata and N. Miyoshi, "Spatial stochastic models for analysis of heterogeneous cellular networks with repulsively deployed base stations," *Performance Evaluation*, vol. 78, pp. 7–17, 2014.

[30] A. Busson, L. Zitoune, V. Vèque, and B. Jabbari, "Outage analysis of integrated mesh LTE femtocell networks," in *IEEE Global Communications Conference GLOBECOM*, Austin, United States, Dec. 2014.

[31] I. L. Cherif, L. Zitoune, and V. Vèque, "The r-l square point process : The effect of coordinated multipoint joint transmission," in *International Wireless Communications & Mobile Computing Conference (IWCMC)*, Dubrovnik, Croatia, Aug. 2015.

[32] J.-M. Kelif, M. Coupechoux, and P. Godlewski, "A fluid model for performance analysis in cellular networks," *EURASIP Journal on Wireless Communications and Networking*, 2010.

[33] "Guidelines for evaluation of radio interface technologies for IMT-advanced," Report ITU-R M.2135-1, Tech. Rep., 2009.

AMONET: A Method for Detecting and Mitigating the Data Rate Degradation due to Interference Over Wireless Networks

Guohao Lan, Sangyup Han, Il-Gu Lee, and Myungchul Kim
School of Computing, KAIST, Daejeon, Republic of Korea
{guohao, ilu8318, iglee9, mck}@kaist.ac.kr

***Abstract*—Recently, wireless networking technologies have been evolving to support wider bandwidth, and longer radio range in denser networks. Therefore, there is a high probability that two or more networks will overlap and result in more co-channel interferences. To mitigate the interference, the centralized network system is a promising solution which is based on the conflicts information provided by the interference detection methods. However, in this paper, we reveal that the existing Passive Interference Detection method (PIE) is not accurate and may cause dramatic throughput decrease in dynamically interfered networks because it is based on a single data rate criterion. Moreover, we propose and implement AMONET, which is a centralized detection method considering the data rate degradation due to interference (DRDI). Our simulation results demonstrate that the proposed scheme can improve aggregate throughput by 2.68x gains in the interfered wireless links over distributed coordination function (DCF), while PIE achieves 1.8x gains over DCF.**

I. Introduction

Within the last decades, we have witnessed the rapid growth of the IEEE 802.11 based wireless networks, as known as Wi-Fi. According to a recent report [1], in the year of 2018, the Wi-Fi traffic will account for 61 percent of the IP traffic and 76 percent of the Internet traffic. This trend of traffic growth results from the increase of mobile station (MS) and the dense deployment of Wi-Fi Access Point (AP). And we can expect that, the current Wi-Fi networks are becoming denser. Unfortunately, due to the limited number of available Wi-Fi channels, a dense network will cause more co-channel interference and result in lower system throughput [2].

To optimize the performance of the Wi-Fi networks, many works have been proposed, such as utilizing multiple antennas [10], [11], controlling transmit power to adjust the interference range of mobile stations [3], [4], applying channel assignment to mitigate the co-channel interference between adjacent APs [5], [6], or using centralized system to schedule packet transmissions [7]–[9]. Other than the adoption of multiple antennas, the other works assume the existence of an accurate data structure which provides the information of interference relations between links in the wireless network. Such a data structure is known as the Conflict Graph (CG) [13]. In the literature, methods for building the conflict graph for a given wireless network can be briefly classified into either active or passive methods. Active interference estimation methods such as, the Interference maps [14] and Micro-probing [15] use active probing to infer the interference relation. Although such approaches are accurate in identifying the interference, their measurement and computing overheads are large when they are applied to a network containing a large number of links. In addition, active methods can hardly be efficient in dynamic environment, in which the network topology changes frequently [16].

On the other hand, the passive methods infer the interference relation mainly based on the passively collected packet traces (either using the off-line packet traces [17], or using on-line monitoring data [16]). Although passive methods have been proven to be efficient to create the conflict graph, they are not accurate in detecting the Data Rate Degradation due to Interference (DRDI), which forces a victim wireless link to degrade its transmission data rate to endure the interference. For instance, the Passive Interference Estimation (PIE) [16] infers the links' interference relation by using the Link Interference Ratio (LIR) [18] at a single data rate only. However, current Wi-Fi networks provide various Modulation and Coding Schemes (MCSs) to enable the trade-off between the transmission efficiency and link reliability. While a high data rate is more efficient in data transmission, a low data rate is more robust against channel noise and co-channel interference. Accordingly, APs can apply the data rate adaptation algorithm [19] to adjust the link's transmission data rate depending on the interference level. When a link is suffering the DRDI, though no packets can be successfully delivered at a high data rate, it can still maintain a high packet delivery rate using a low data rate. In this circumstance, the existing passive detection methods [16], which using the LIR value calculated at a single data rate, can't detect the DRDI due to the dynamic rate adaptation. However, an MS with a lower data rate requires more time to transmit a packet than the MSs with higher data rates. Thus, if there is one 'slow' MS contending the wireless channel with some 'fast' MSs, it will induce the well-known performance anomaly problem in 802.11 networks [20], such that the throughput of all the 'fast' mobile stations will degrade as low as the 'slow' one. Therefore, without precisely detecting the DRDI, existing passive interference estimation methods are

ISBN 978-3-901882-83-8

unreliable. In this paper, we present AMONET to solve the shortcoming in existing passive interference estimation. This paper makes the following contributions:

- Through extensive measurement studies, we demonstrate that the existing work in passive interference estimation performs poorly in dynamically interfered networks and leads to throughput degradation because it cannot detect the DRDI. To address this problem, we propose AMONET: a passive interference estimation method to detect and mitigate the DRDI.
- We implement the AMONET in QualNet [21] and integrat it to a centralized scheduling system, the Centaur [7]. We compare the improvement in system throughput of Centaur when using the conflict information from AMONET and PIE. The results indicate that AMONET can achieve a system throughput gain of 1.30× and a throughput gain of 2.60× on the victim links, over the Distributed Coordination Function (DCF) while providing better link fairness, in the dense wireless network. In comparison, PIE can achieve a throughput gain of 1.12× and 1.80× over DCF, respectively.

The rest of this paper is organized as follows. In Section II, we discuss related works in passive interference estimation. In Section III, we present the experiment result in a real testbed, which shows the throughput degradation due to the DRDI. And, we present our measurement study to demonstrate the shortcomings in current passive interference estimation. In Section IV, we present the detailed design of AMONET. In Section V, we show a comprehensive evaluation of AMONET by simulation. We conclude our work in Section VI.

II. Related work

There are many works that focus on interference measurement in the network research community, either active or passive methods. In this section, we briefly survey the related works in the branch of passive interference estimation.

A. Existing approaches in passive interference estimation

Earlier studies in this field, such as Jigsaw [17] and WIT [22] collect off-line data to analyze network performance. WIT analyzes the performance of the IEEE 802.11 MAC protocol using the data traces collected by several sniffers. Similarly, Jigsaw also uses sniffers to collect data traces to study link performance. The most recent study of the passive method is the PIE. PIE is a centralized system which infers the interference relation across the entire Wi-Fi networks using the data traces collected at different APs. A central controller is used to merge the traffic traces and analyze the LIR (will be described in Section 2.2) and packet overlapping relation, then the controller can accurately infer the APs' Carrier Sense and Hidden Terminal relation.

B. The Link Interference Ratio (LIR)

LIR is a widely used metric in current passive interference detection to infer the interference relation between any two links [18], [20]. For a pair of links, LIR is the ratio of the transmission performance when they transmit simultaneously, to the performance when they transmit individually. Its value ranges from 0 to 1. LIR of 1 indicates the two links do not interfere, whereas, LIR of 0 indicates there is heavy interference between the two links. In practice, a threshold of 0.8 is widely used to judge the existence of interference [18], [20]. Note that, we use the notation L_{AP_i,MS_j} to represent a wireless link from AP_i to MS_j.

LIR can be calculated in both passive and active ways. The active way is the Unicast Bandwidth Test (UBT) [18] which uses the measured throughputs of links when they transmit unicast data. The LIR estimated using UBT is given by:

$$LIR\_UBT_{AP_3MS_4 \to AP_1MS_2} = \frac{U_{AP_3MS_4 \to AP_1MS_2}}{U_{AP_1MS_2}} \quad (1)$$

where $U_{AP_1MS_2}$ is the unicast throughput of L_{AP_1,MS_2} when it transmits individually, and $U_{AP_3MS_4 \to AP_1MS_2}$ is the unicast throughput of L_{AP_1,MS_2} when it transmits simultaneously with L_{AP_3,MS_4}. The LIR_UBT directly shows the decrease of transmission efficiency on L_{AP_1,MS_2} when it transmits together with L_{AP_3,MS_4}. UBT is widely regarded as the reference of interference estimation [17], [18], [20]. However, it requires an overhead of $\mathcal{O}(n^4)$ to estimate the interference relation for an *n* node network, which is not applicable in practice. According to the experiment result given in [23], the running time of UBT in a 20 nodes topology is more than one hour.

On the other hand, a passive way to calculate the LIR is to use the overlapping packets delivery rate [16], which shows the probability of a wireless link to successfully transmit its packet at a certain data rate when it transmits simultaneously with the potential interferer. The LIR calculated using the Packet Delivery Rate (PDR) is defined as:

$$LIR\_PDR_{AP_3MS_4 \to AP_1MS_2,r} = \frac{R_{AP_3MS_4 \to AP_1MS_2,r}}{R_{AP_1MS_2,r}} \quad (2)$$

where $R_{AP_1MS_2,r}$ is the packet delivery rate of L_{AP_1,MS_2} when it transmits individually at the data rate *r*, and $R_{AP_3MS_4 \to AP_1MS_2,r}$ is the delivery rate of L_{AP_1,MS_2} when it transmits simultaneously with L_{AP_3,MS_4} at the data rate *r*.

In summary, in addition to detecting the well-studied Carrier Sense and the collision induced interference in the hidden terminal problem, the AMONET is able to detect the Data Rate Degradation which is also caused by the hidden terminal problem.

III. Problem definition and motivation

In this section, we first introduce the types of interference that we are focusing in this paper. After that, we present an experiment in a real testbed to study the DRDI. Then, a measurement study on the interference estimation in a multiple rates environment is presented, in which we analyze the interference detection result of PIE, and demonstrate that the existing passive interference detection methods can't detect the DRDI.

A. Types of interference

In this paper we focus on the interferences among the downlink transmissions, from APs to MSs, because the downlink traffic occupies around 85% of the entire traffic in the Wi-Fi networks [24]. In the downlink case, the interference can be broadly classified into two categories: **1)** the Carrier Sense (CS) interference between two APs, which determines how the APs can share the wireless channel; and **2)** the collision induced interference at the MSs, due to the hidden terminal problem.

For any two APs sharing the same wireless channel, AP_i and AP_j, there are four cases of **CS interference**: **1)** $AP_i \leftrightarrow AP_j$: if AP_i and AP_j can carrier sense the transmission of each other; **2)** $AP_i \leftarrow AP_j$: if AP_j can carrier sense the transmission of AP_i, but AP_i can't sense AP_j; similarly we have **3)** $AP_i \rightarrow AP_j$; **4)** $AP_i \nleftrightarrow AP_j$: if AP_i and AP_j can't carrier sense each other. If at least one of the two APs can't carrier sense the other one, their simultaneous transmissions will cause the collision induced interference on the MSs. This is widely known as the hidden terminal problem. Although the concept of hidden terminal problem is widely used in current literature [13], [15], we still lack of a detailed definition of it in multiple rates scenario, in which APs can still transmit their packets successfully after degrading the data rates, if the interference from the hidden terminal is not severe. Note that a wireless link may degrade its transmission data rate due to various reasons, such as the increase of transmission distance, the channel noise, or the interference from non-Wi-Fi devices. The problem we are focusing in this paper arises from the collision induced interference due to the hidden terminal problem. We define and use the following definitions in this paper:

Definition 1. Hidden Terminal Interference (HTI). If 1) the simultaneous transmission of L_{AP_3,MS_4} causes packets collisions on MS_2 and 2) the Packet Delivery Rate of L_{AP_1,MS_2} is below 0.8 (more than 20% packet loss) even if L_{AP_1,MS_2} has degraded its transmission data rate to the lowest one. Then, we say L_{AP_3,MS_4} causes the HTI on L_{AP_1,MS_2}.

Definition 2. Data Rate Degradation due to Interference (DRDI). If 1) the concurrent transmission of L_{AP_3,MS_4} causes packets collisions on MS_2 but 2) L_{AP_1,MS_2} can still achieve a Packet Delivery Rate over 0.8 by using a lower data rate. Then, we say L_{AP_3,MS_4} causes the DRDI on L_{AP_1,MS_2}.

Both HTI and DRDI can be regarded as the symptoms of the hidden terminal problem, but they will cause different results.

B. Experiments in real testbed

We set up an indoor testbed to investigate the significant throughput degradation caused by the DRDI in a practical Wi-Fi networks. As shown in Fig. 1, the testbed consists of two Cisco WRT610N series APs, AP_1 and AP_2, that have wired connection to the PCs running the Iperf [25]. We have three

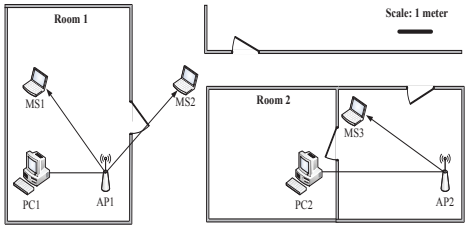


Fig. 1: The topology of the testbed consists of two APs and three mobile stations.

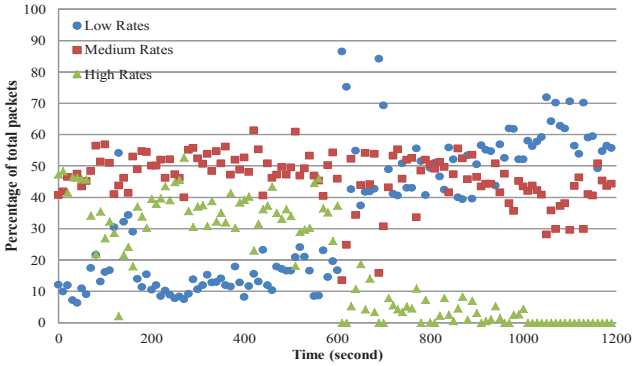


Fig. 2: Percentages of packets transferred from AP_1 to MS_2 at different data rate groups during the 20 minutes experiment.

mobile stations, MS_1 is a laptop with the Realtek RTL8188CE wireless card, MS_2 is a laptop with the Ralink RT3090 card and MS_3 is a laptop using the Intel Dual Band Wireless-AC 7260 wireless card. MS_1 and MS_2 are associated with AP_1 and MS_3 is associated with AP_2. The APs are configured to use the same channel in the 2.5GHz band. Moreover, we fixed the network mode as IEEE 802.11n and used the default auto-rate adaptation algorithm [19] in the APs to support data rate adaptation. In all the experiments we disabled the RTS/CTS handshake and fixed the transmission power to ensure those two APs can't carrier sense each other. We put MS_1 inside Room 1 which was three meters away from AP_1, so that it would not suffer any interference from the transmission of AP_2. On the other hand, we put MS_2 at the corridor between Room 1 and Room 2, and it was also three meters away from AP_1. But the transmission from AP_2 would cause the DRDI on MS_2. We used the Iperf running on both PC_1 and PC_2 to generate the downlink UDP traffic from APs to mobile stations. The offered traffic load was configured as 30 Mbps.

We set up a 20 minutes experiment. In the first 10 minutes, only AP_1 was activated, after that we also activated AP_2, so that the transmission from AP_2 to MS_3 would cause the DRDI on MS_2. During the experiment, we collected the traces of the wireless network traffic and analyzed the adaptation of the data rates. Because of the rate diversity, we simply classify the data rates into three groups: the data rates of 6.5, 13 and 19.5

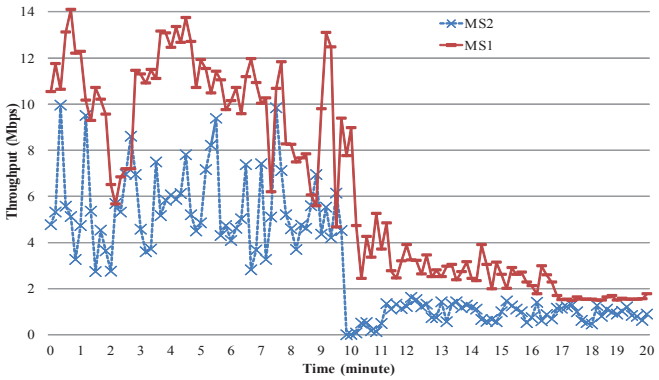


Fig. 3: The throughputs of both MS_1 and MS_2 during the 20 minutes experiment.

Mbps serve as the Low Rates group; 26 and 39 Mbps serve as the Medium Rates group; 52 Mbps and higher rates serve as High Rates group. Fig. 2 shows the percentages of packets transferred from AP_1 to MS_2 using each of those three data rate groups during the 20 minutes experiment. We can see that in the first 600 seconds, the percentage of the packets transmitted using 'High' data rates is about 50 percent, and only around 10 percent of packets were transmitted using a 'Low' data rate. However, after 600 seconds, when the DRDI happened, the percentage of High Rates decreased below 10 percent, whereas, the percentage of Low Rate increased and was above 50 percent. This data rate degradation happened on MS_2 caused the performance anomaly problem on MS_1. As shown in Fig. 3, in the first 10 minutes, the throughputs of MS_1 and MS_2 are around 10 Mbps and 6 Mbps, respectively. After 10 minutes, the throughput of MS_2 decreased to 1 Mbps due to the DRDI from AP_2, and the throughput of MS_1 also decreased to 3 Mbps due to the performance anomaly problem caused by the DRDI on MS_2. Thus, it is essential to detect and provide the DRDI information to the upper-layer applications, such as the centralized scheduling system, to resolve the interference. In the next section, we present our measurement study of the existing passive interference estimation method, which shows that the existing method can't accurately detect the DRDI.

C. A measurement study of passive interference estimation in dynamic environments

Previous studies [4], [23] have shown that the interference relation between wireless links is affected by the link's physical data rate. Although previous study on passive interference estimation, PIE, has been proven to be accurate in estimating the LIR_PDRs for different data rates, it only uses the LIR_PDR calculated at a single data rate to judge the interference relation. Without showing how to apply the LIR_PDRs calculated at different data rates to infer the interference relations in multi-rates environment, it may result in a misjudgement of interference relation.

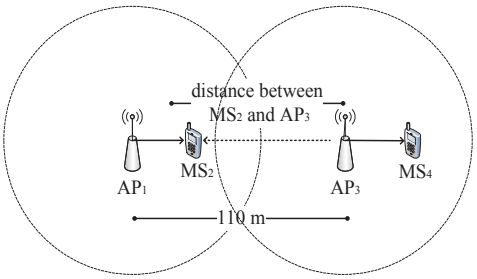
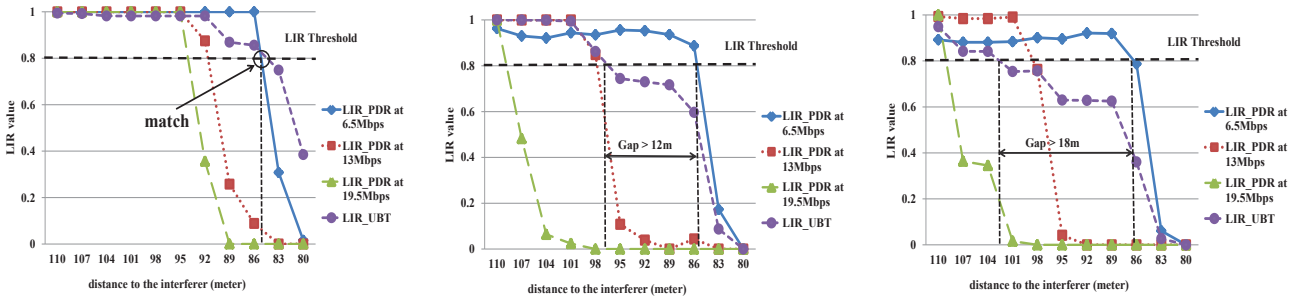


Fig. 4: A simple Hidden Terminal topology consists of two 802.11n wireless links.

To understand the above problem, we conduct extensive simulations in QualNet using a simple hidden terminal topology shown in Fig. 4. The topology consists of two APs and two MSs sharing the same channel in 802.11n. We use the Two-ray pathloss model and the lognormal shadowing model with 8dB shadowing mean in all the simulations in this paper. The solid lines with arrow represent the association relations between AP and MS, while the dashed line with arrow represents the potential interference relation. Transmission ranges of the APs are configured as 80 meters. The dash circles indicate the carrier sensing ranges of AP_1 and AP_3. The distance between AP_1 and AP_3 is around 110 meters and they can't sense each other. Thus, those two links are in the hidden terminal problem and the transmission of L_{AP_3,MS_4} may cause interference on L_{AP_1,MS_2}. However, according to our previous definitions, depending on the distance between MS_2 and AP_3 and the offered traffic load, it may result in either HTI or DRDI.

We construct a number of scenarios by adjusting the distance from MS_2 to its potential interferer AP_3. We estimate the interference level of L_{AP_3,MS_4} on L_{AP_1,MS_2} for every three meters when the distance changes from 110 meters to 65 meters. In addition, we adjust the offered traffic load on those two links among 3, 6 and 9 Mbps, to take the influence of traffic load into account. Every scenario was simulated more than 20 times by varying the random seed value used in the QualNet, which will affect the feature of both signal propagation and wireless environment. We also disable the Request-to-Send/Clear-to-Send (RTS/CTS) handshake for all the nodes, and apply the Auto Rate Fallback (ARF) implemented in the QualNet to support rate adaptation. We use both unicast throughput and packet delivery rate, defined in Eq. (1) and Eq. (2), to calculate the LIR_UBT and LIR_PDR, respectively. Note that, in case of deciding interference relation, the LIR_Threshold of 0.8 is applied. If the LIR_UBT is less than the LIR_Threshold, the UBT [18] will infer the links are in an interference relationship. Similarly, if the LIR_PDR of the lowest data rate (6.5 Mbps in case of 802.11n, notated as $\text{LIR\_PDR}_{6.5Mbps}$) is less than LIR_Threshold, PIE will judge them as interference. Like the previous works in the literature [18], [20], we regard the interference relation measured by the UBT as the ground truth.

Fig. 5 shows the distributions of the LIRs of L_{AP_3,MS_4}



(a) Light traffic load: 3 Mbps. (b) Medium traffic load: 6 Mbps. (c) Heavy traffic load: 9 Mbps.

Fig. 5: The LIRs of L_{AP_3,MS_4} on L_{AP_1,MS_2} when MS_2 moves forward to AP_3.

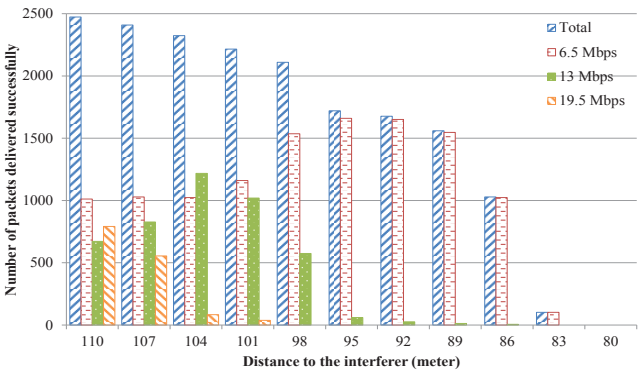


Fig. 6: The number of packets successfully delivered by L_{AP_1,MS_2} at three different data rates (6.5, 13 and 19.5 Mbps) under the interference from L_{AP_3,MS_4} and with a 6 Mbps traffic load.

on L_{AP_1,MS_2}. Three data rates (6.5, 13 and 19.5 Mbps) are used by L_{AP_1,MS_2} during the experiment. Thus, we estimate the LIR_PDR based on the packet delivery rate of L_{AP_1,MS_2} at each of those data rates (following Eq. (2)), denoted as LIR_PDR$_{6.5Mbps}$, LIR_PDR$_{13Mbps}$ and LIR_PDR$_{19.5Mbps}$, respectively. Fig. 5 also shows the LIR_UBT calculated based on the unicast throughput of L_{AP_1,MS_2} (following Eq. (1)). As we expected, when MS_2 moves closer to AP_3, all the three LIR_PDRs and LIR_UBT are decreased. Moreover, the decrease of LIRs (both LIR_PDR and LIR_UBT) start earlier in the case when the offered traffic load is heavy (9 Mbps), than in the cases when traffic load is either light (3 Mbps) or medium (6 Mbps). This is because, with the increase of the offered traffic load, the collision happens more frequently on MS_2. **In case of light traffic load**, shown in Fig. 5(a), the LIR_UBT indicates that L_{AP_1,MS_2} suffers interference from L_{AP_3,MS_4} when the distance between MS_2 and AP_3 is less than 83 meters. Moreover, the LIR_PDR$_{19.5Mbps}$ and LIR_PDR$_{13Mbps}$ decrease below the LIR_Threshold much earlier than LIR_UBT, due to their poor robustness against the interference. However, the value of LIR_PDR$_{6.5Mbps}$ decreases below the LIR_Threshold not until the distance reduces to 83 meters. In this case, the estimation result of PIE using the LIR_PDR$_{6.5Mbps}$ matches the result of the UBT which we regarded as the reference of interference. Both of them infer the interference on L_{AP_1,MS_2} starts when the distance is less than 83 meters. **In case of medium traffic load**, shown in Fig. 5(b), with the increasing of the traffic load, more collisions happen at MS_2. Thus, the LIR_UBT shows that L_{AP_3,MS_4} will cause heavy interference on L_{AP_1,MS_2} from a position much earlier than that in the light traffic load case. From Fig. 5(b), we can see that the LIR_UBT drops below the LIR_Threshold when the distance is around 95 meters. However, LIR_PDR$_{6.5Mbps}$ still maintains above the LIR_Threshold until the distance decreases to 83 meters. The UBT estimates that L_{AP_1,MS_2} suffers interference from L_{AP_3,MS_4} when the distance is 95 meters, whereas, the PIE shows the interference starts from 83 meters, which is a 12 meters difference. Thus, there is a gap area in which the estimation results of the UBT and PIE are different. **In case of heavy traffic load**, as shown in Fig. 5(c), the result is similar to that in the medium traffic load. We can notice that the LIR_UBT drops below the LIR_Threshold when the distance is around 101 meters. But, the value of LIR_PDR$_{6.5Mbps}$ can maintain above the LIR_Threshold until the distance is smaller than 83 meters. In this case, the range of the gap area increases to 18 meters.

An interesting question is: *what causes the gap between the estimation results between the UBT and PIE? And what happens on the link L_{AP_1,MS_2} within the gap area?*

To answer the above question, let's analyze the gap area shown in Fig. 5(b) and the measurement result of L_{AP_1,MS_2} shown in Fig. 6, which shows the number of successful packets delivered at each of the data rates that L_{AP_1,MS_2} used. In Fig. 5(b), the gap area starts from 95 meters to 83 meters: the LIR_UBT drops below the LIR_Threshold when the distance is around 95 meters, whereas, the LIR_PDR$_{6.5Mbps}$ drops below the LIR_Threshold not until 83 meters. Fig. 6 shows that the total number of successful packets drops dramatically when the distance decreases from 98 meters to 95 meters. Moreover, when the distance is between 95 meters and 86 meters (the gap area), nearly all the successful packets are sent at the data rate of 6.5 Mbps. Again, this is because AP_1 de-

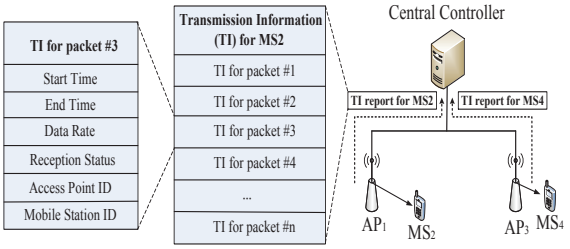


Fig. 7: An example of the AMONET architecture which consists of a central controller, two APs and two MSs.

grades its transmission data rate to resist the interference from L_{AP_3,MS_4}. According to the distribution of LIR_PDR$_{6.5Mbps}$ shown in Fig. 5(b), L_{AP_1,MS_2} can still achieve a high packets delivery rate at 6.5 Mbps until the distance reaches 83 meters. However, due to the decrease of total successful packets and the degradation of transmission data rate, the throughput of link L_{AP_1,MS_2} decreases dramatically within the gap area, as the LIR_UBT indicates in Fig. 5(b). The interference L_{AP_3,MS_4} on L_{AP_1,MS_2} within the gap area is the DRDI which we defined previously. Lastly, when the distance is smaller than 83 meters, even the lowest data rate can't resist the interference, both the LIR_UBT and LIR_PDR$_{6.5Mbps}$ decrease to 0. From Fig. 6, we can notice that nearly no packets can be received at MS_2 when the distance is smaller than 83 meters. In this case, the interference L_{AP_3,MS_4} on L_{AP_1,MS_2} is the HTI. Note that, using the LIR_PDR at a single rate in PIE can detect the HTI but fails to detect the DRDI. Because the DRDI happens in the gap area as shown in Fig. 5, in which the LIR_PDR$_{6.5Mbps}$ is still above the LIR_Threshold. On the other hand, though the UBT can clearly detect both the HTI and DRDI using the LIR_UBT, it is not applicable in practice due to its large computing overhead, as mentioned before.

IV. The design of AMONET

In this section, we present the design of AMONET. We outline the architecture of AMONET and introduce the algorithms to detect the Carrier Sense, HTI, and DRDI.

A. System overview

Following the trend in the existing passive interference estimation, AMONET is a centralized system consisting of a central controller, and with all the APs connected to the controller through wired networks, where the main function of interference estimation is implemented. The APs are required to record the Transmission Information (TI) for every packet they have transmitted. The information includes: **1)** an accurate timestamp showing the start time and end time of the packet; **2)** the data rate the packet has been transmitted; **3)** the reception status showing if the packet has been successfully delivered; and **4)** the IDs of the AP (source) and MS (destination). As an example shown in Fig. 7, for every packet AP_1 sends to MS_2, AP_1 creates a timestamp for that packet and records the data rate it used for transmission. In order to achieve this, AMONET requires a slightly modification on the APs. After receiving the corresponding ACK from MS_2, AP_1 will update the packet's reception status. APs will report the collected TIs to a central controller periodically. The length of the report period can be configured empirically. While a short period can ensure a lower computation overhead and faster update, a long period can enable higher detection accuracy. In our setting, the report period is configured as three seconds, which ensures both low computation overhead and high detection accuracy.

B. Detection of the Carrier Sense (CS) interference

Corresponding to each of the four Carrier Sense relations we introduced in Section 3, for any two APs, AP_i and AP_j, their packet transmissions can have four kinds of **overlapping relation**: **1) Non-Overlapping**: if their CS relation is $AP_i \leftrightarrow AP_j$, according to the IEEE 802.11 CSMA/CA mechanism, both of the two APs will defer their transmissions to avoid packet collisions whenever they sense the other AP is sending packets. Thus, the controller will not find any overlapped packet between those two APs; **2) One-way-Overlapping**: in case of $AP_i \leftarrow AP_j$, the controller can observe that the packets from AP_i overlap with the packets from AP_j. Because AP_i will keep sending its packets even if AP_j is using the channel. However, AP_j will defer its transmission while AP_i is occupying the channel. Similarly, we have the relation $AP_i \rightarrow AP_j$; **3) Two-way-Overlapping**: if $AP_i \nleftrightarrow AP_j$, neither AP_i nor AP_j will not defer its transmission even if the other AP is sending, thus the controller will observe their packets overlap with each other.

To infer the CS interference, the controller will merge and sort the collected TIs in the order of the transmission start-time based on the same timeline. Note that, this requires the times at both the APs and controller are synchronized, which can be achieved using the Precision Time Protocol [26]. After sorting, the controller analyzes the overlapping relation between the APs' packet transmissions. Then, based on the packets overlapping relationship, the controller can infer the APs' Carrier Sense relation.

The details are shown in **Algorithm 1**. The controller takes merged and sorted TI list, $LIST_{TI}$, as the input and analyze their overlapping relation. In order to check the overlapping relation of two TIs from two different APs, TI_i and TI_j, the controller compares their timestamps. If the start-time of TI_i (denoted as TS_i) is later than the start-time of TI_j (TS_j), but earlier than TI_i's end-time (TE_i), we infer that TI_j overlaps with TI_i. On the other hand, if $TS_j < TS_i < TE_j$, we can infer that TI_i overlaps with TI_j. For every two TIs in the $LIST_{TI}$, the controller analyzes their overlapping relation and records the overlapping times into the result set, $LIST_{overlap}$. Then, using the records in the $LIST_{overlap}$, the controller can judge the CS relation between two APs, AP_i and AP_j. As recorded in $LIST_{overlap}$, if the number of packets from AP_i overlapped the packets from AP_j is more than the threshold δ (equals to 50 in our configuration to assure the accuracy),

Algorithm 1 Detection of the Carrier Sense (CS) Interference

Require: **1)** $LIST_{TI}$; **2)** $LIST_{AP}$.
Ensure: **1)** CS_Result; **2)** $LIST_{overlap}$.
function CSRELATION($LIST_{TI}, LIST_{AP}$)
  $LIST_{overlap} \leftarrow empty$, $CS\_Result \leftarrow empty$
  for any TI_i and TI_j **in** $LIST_{TI}$ **do**
    if $TI_i.AP_{ID} \neq TI_j.AP_{ID}$ **then**
      CHECKOVERLAP($TI_i, TI_j, LIST_{overlap}$)
  for any AP_i and AP_j **in** $LIST_{AP}$ **do**
    if $!LIST_{overlap}.Time(AP_i, AP_j) > \delta$ **then**
      $CS\_Result.Add(AP_i, AP_j)$
  return CS_Result

then the controller infers that AP_i can't carrier sense AP_j, otherwise, AP_i can carrier sense AP_j. Finally, the detection result of CS relation is recorded in the result set, CS_Result.

C. Detection of the HTI and the DRDI

The prerequisite for any two links in the HTI or DRDI relation is that at least one of APs can't carrier sense the other one. Therefore, to detect the links' interference relation, the controller gets the Carrier Sense relations between all the APs, CS_Result, from **Algorithm 1**. The pseudo code for detecting the HTI and DRDI is given in **Algorithm 2**. The controller maintains a list which stores the information for all the links, $LIST_{link}$. For any two links in $LIST_{link}$, $Link_i$ and $Link_j$, the controller checks APs' Carrier Sense relation by using CS_Result. If their APs can't carrier sense each other, the controller will analyze their interference relation based on LIR_PDR. The algorithm for calculating the links' LIR_PDR is given in **Algorithm 3**.

Based on Eq. (2), for two links, $Link_i$ and $Link_j$, $LIR\_PDR_{i\to j,r}$ shows the performance loss on $Link_j$ when it transmits at data rate r simultaneously with $Link_i$. As shown in **Algorithm 3**, $R_{i\to j,r}$ indicates the packet delivery rate of $Link_j$ at rate r when transmitting together with $Link_i$. To calculate $R_{i\to j,r}$, the controller measures the overlapping information between those two links. The value of $O_{i\to j,r}$ is the number of packets from $Link_j$ transmitted at rate r overlapped by the packets from $Link_i$. Correspondingly, $OL_{i\to j,r}$ is the number of overlapping packet loss at $Link_j$ when overlap with the transmission from $Link_i$. Similarly, the controller can measure the $R_{j,r}$, which is the packet delivery rate of $Link_j$ at rate r when transmitting without $Link_i$. $I_{j,r}$ is the number of packet from $Link_j$ transmitted at rate r isolated with the transmission from $Link_i$. Accordingly, $IL_{j,r}$ is the number of packet lost among those isolated transmission. Then, based on $R_{i\to j,r}$ and $R_{j,r}$ we can get $LIR\_PDR_{i\to j,r}$. The controller will repeatedly calculate $LIR\_PDR_{i\to j,r}$ for every data rate r used by $Link_j$ lately. LIR_PDR will be used in **Algorithm 2** to compute the interference relation between two links.

Similar with PIE, the accuracy of AMONET in interference detection is affected by the network scale and transmission diversity. In a large network consisting of multiple interferers, the accuracy of validating the interferer depends on a high transmission diversity. In order to identify the actual interferer from a group of potential interferers, AMONET needs to calculate the exact packet delivery rates of the victim link when transmitting together and isolated with each of the potential interferers, which assumes that the transmission of different links should not highly overlapped. AMONET is able to identify the actual interferer when the transmission overlaps between two links are less than 60%. In practice, the measurement result [17] achieved in a real WLAN shows that around 90% of the wireless transmissions overlap less than 20% of the time. This transmission diversity enables AMONET to be accurate in the real environment.

Algorithm 2 Detection of the HTI and DRDI

Require: **1)** $LIST_{Link}$; **2)** $LIST_{overlap}$; **3)** CS_Result.
Ensure: $Interference\_Result$.
function DETECTIONIR($LIST_{Link}$)
  for any $Link_i$ and $Link_j$ **in** $LIST_{Link}$ **do**
    $AP_i \leftarrow Link_i.AP$, $AP_j \leftarrow Link_j.AP$,
    if $!CS\_Result.Has(AP_i, AP_j)$ **then**
      $Relation_{i\to j} \leftarrow$COMPUTEIR($Link_i, Link_j$)
      $Interfernce\_Result.Add(Relation_{i\to j})$
  return $Interfernce\_Result$
function COMPUTEIR($Link_i, Link_j$)
  $LIR\_PDR_{i\to j} \leftarrow$ COMPUTELIR($Link_i, Link_j$)
  /*ComputeLIR is given in Algorithm 3*/
  if $LIR\_PDR_{i\to j}[0] < Threshold$ **then** HTI
  if $LIR\_PDR_{i\to j}[1] < Threshold$ **then** $DRDI$
  if $LIR\_PDR_{i\to j}[2] < Threshold$ **then** $DRDI$

Algorithm 3 Calculate the LIR

Require: **1)** $LIST_{Link}$; **2)** $LIST_{overlap}$; **3)** CS_Result.
Ensure: $LIR\_PDR_{i\to j}$.
function COMPUTELIR($Link_i, Link_j$)
  $Overlap_{i\to j} = LIST_{overlap}.Get(Link_i, Link_j)$
  for any r **in** $Overlap_{i\to j}.Rate()$ **do**
    $R_{j,r} \leftarrow 1 - IL_{j,r}/I_{j,r}$
    $R_{i\to j,r} \leftarrow 1 - OL_{j,i,r}/O_{j,i,r}$
    LIR_PDR$_{i\to j}[r] = R_{i\to j,r}/R_{j,r}$
  return LIR_PDR$_{i\to j}$

V. EVALUATION OF AMONET

In this section, we conducted extensive simulations in the QualNet simulator in order to provide that the interference information provided by AMONET can benefit the centralized scheduling algorithm more than previous works.

A. Experimental setup

We compare AMONET with the PIE and integrate both of them to the centralized scheduling algorithm, Centaur [7]. Using the conflict information provided by either AMONET or PIE, Centaur can avoid the downlink interference by allocating conflicted links to non-overlapping time slots, and thus improve the aggregate throughput. Different from AMONET, PIE can't detect the DRDI. Thus, the system throughput of Centaur will decrease due to the performance anomaly

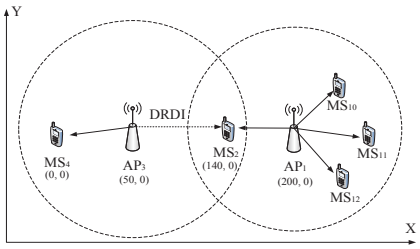


Fig. 8: A two-APs scenario consists of five wireless links. The distance between two APs is 150 meters, and the distance between AP_3 and MS_2 is 90 meters.

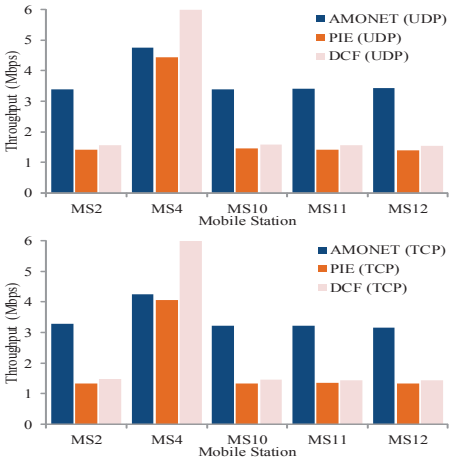


Fig. 9: The throughput of the mobile stations in the Two-APs topology with 6 Mbps traffic load.

problem induced by the DRDI. We evaluate the performance of AMONET in both simple and complex scenarios. First, we use a simple scenario to show the DRDI can cause the aggregate throughput decrease. Second, we use a more complex scenario includes both HTI and DRDI. We evaluate the performance of all scenarios in terms of the aggregate throughput and fairness. The well-known Jain's fairness index [27] is used to evaluate the fairness, the fairness index with a value of 1 indicates the system is 100% fair, whereas, a value of 0 indicates unfairness.

Note that, in all scenarios, we employ IEEE 802.11n and enable the ARF and with RTS/CTS disabled. We assume the APs have the same transmission range and are configured to send saturated traffic to their MSs. Both TCP and UDP traffic are used in all scenarios, with the packet size fixed at 1,400 bytes. The experiments are repeated more than 25 times using different random seeds in the QualNet, that will affect the characteristics of the traffic and wireless environment.

B. Simple scenario: two-APs topology with the DRDI only

In this section, we construct a simple two-APs scenario which only includes the DRDI. We evaluate the performance using both UDP and TCP traffic, with a 6 Mbps traffic load for each mobile station. As shown in Fig. 8, AP_1 and AP_3

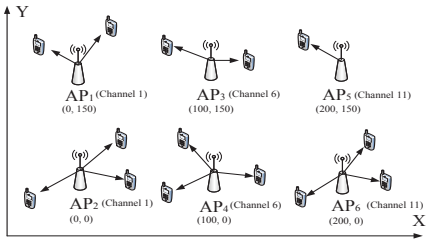


Fig. 10: A complex scenario consists of 6 APs and 14 MSs.

TABLE I: Normalized system throughput gains of Centaur integrate with AMONET and PIE over DCF and the system fairness.

Traffic	Method	Gains	Fairness Index
UDP	AMONET	1.30×	0.91
UDP	PIE	1.12×	0.86
TCP	AMONET	1.28×	0.91
TCP	PIE	1.12×	0.85

can't carrier sense each other, and the transmission from AP_3 will cause the DRDI on link L_{AP_1,MS_2}.

Simulation results. Fig. 9 shows the throughput for each of the MSs, due to the DRDI caused by the interference from link L_{AP_3,MS_4} on link L_{AP_1,MS_2}, the throughput of MS_2 is less than 2 Mbps. Moreover, the performance anomaly problem induced by the rate degradation affected all the other three MSs that associate to AP_1, and made the throughputs of those MSs are as low as that of MS_2. In case of Centaur integrated with PIE (PIE-Centaur), because PIE can't detect the DRDI, Centaur performs no better than DCF. On the other hand, when integrated with AMONET, Centaur can allocate different time slots to the transmissions of L_{AP_3,MS_4} and L_{AP_1,MS_2}, which mitigates the interference occurs on MS_2. As a result, L_{AP_1,MS_2} will be free from the DRDI from L_{AP_3,MS_4}, and thus, will increase the throughput on all the other MSs.

C. Complex scenario: multi-APs topology with both HTI and DRDI

In this subsection, we evaluate AMONET in a complex scenario which includes both HTI and DRDI. As shown in Fig. 11, our topology consists of six APs and their positions are denoted as (x, y). Besides, every two adjacent APs use one of the three orthogonal channels (Channel 1, 6 and 11) in 2.4 GHz, and with a distance of 100 or 150 meters, so that they are out of the carrier sensing range of each other. We distribute 14 MSs and associate them to each of the APs, as shown in Fig. 10, each of the APs may have one to three MSs. Note that, in each of the scenarios we used, the ratio of links suffering either the HTI or DRDI is about 25%, which means around four links among those 14 links are suffering one of those two interferences. This ratio matches the measurement result presented in [16], that about 10% to 30% of links in a Wi-Fi network suffering those two interferences.

Simulation results. Table 2 shows the throughput gains of Centaur over DCF for the complex scenario, when using

TABLE II: Aggregate throughput gains of Centaur with AMONET and PIE on the interfered wireless links over DCF.

Traffic	Mechanism	Gains
UDP	AMONET	2.68×
UDP	PIE	1.80×
TCP	AMONET	2.59×
TCP	PIE	1.82×

the interference information generated by AMONET and PIE, respectively. Because the scenario consists of both HTI and DRDI, the conflict graph created by PIE is not accurate enough to help Centaur to avoid the DRDI. However, comparing with DCF, PIE-Centaur can improve the aggregate throughput by avoiding the HTI and achieve a throughput gain of 1.12× over DCF. On the other hand, using the HTI and DRDI information provided by AMONET, Centaur can effectively resolve both of those conflicts by allocating the conflicted transmissions into different transmission timeslots. As shown in the results, AMONET-Centaur can achieve a throughput gain of 1.30× over DCF. Table 3 shows the aggregate throughput gains of Centaur on the interfered links. The results indicate that, with AMONET, Centaur can achieve a throughput gain more than 2.50 on those interfered AP-to-MS links over DCF, while with PIE, Centaur can only achieve a 1.80 gain over the DCF. This clearly shows that Centaur can largely increase the throughput of the interfered links by using AMONET. In addition, in all the cases, Centaur can achieve better fairness when using AMONET, compare with PIE and DCF.

VI. Conclusion and Future Work

In this paper, we presented a detailed measurement study of the passive interference estimation in multi-rate Wi-Fi networks. We found that the existing works are not able to detect the DRDI, which will cause dramatic throughput degradation. To address this problem, we presented the AMONET and integrated it with a centralized scheduling system to detect and mitigate the interferences. The results showed that AMONET can greatly benefit the centralized scheduling system and achieved higher throughput than the previous works in interference estimation. As a part of our future work, we will implement our AMONET system in real testbed to evaluate its performance and accuracy in interference detection.

VII. Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Human Resource Development Project for Brain scouting program (IITP-H7106-15-1011) supervised by the IITP (Institute for information & communications Technology Promotion).

References

[1] Cisco, "The zettabyte era: trends and analsis," 2015.

[2] M. A. Ergin, K. Ramachandran, and M. Gruteser, "Understanding the effect of access point density on wireless lan performance," in *Proceedings of ACM MobiCom*, 2007, pp. 350–353.

[3] V. P. Mhatre, K. Papagiannaki, and F. Baccelli, "Interference mitigation through power control in high density 802.11 wlans," in *Procceedings of IEEE INFOCOM*, 2007, pp. 535–543.

[4] J. Huang, G. Xing, and G. Zhou, "Unleashing exposed terminals in enterprise wlans: A rate adaptation approach," in *Proceedings of IEEE INFOCOM*, 2014, pp. 2481–2489.

[5] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh, "A client-driven approach for channel management in wireless lans," in *Proceedings of IEEE INFOCOM*, 2006, pp. 1–12.

[6] E. Rozner, Y. Mehta, A. Akella, and L. Qiu, "Traffic-aware channel assignment in enterprise wireless lans," in *Proceedings of IEEE ICNP*, 2007, pp. 133–143.

[7] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "Centaur: realizing the full potential of centralized wlans through a hybrid data path," in *Proceedings of ACM MobiCom*, 2009, pp. 297–308.

[8] J. Manweiler, N. Santhapuri, S. Sen, R. R. Choudhury, S. Nelakuditi, and K. Munagala, "Order matters: transmission reordering in wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 2, pp. 353–366, 2012.

[9] W. Zhou, D. Li, K. Srinivasan, and P. Sinha, "Domino: relative scheduling in enterprise wireless lans," in *Proceedings of ACM CoNEXT*, 2013, pp. 381–392.

[10] D. S. Chan, T. Berger, and L. Tong, "Carrier sense multiple access communications on multipacket reception channels: theory and applications to ieee 802.11 wireless networks," *Communications, IEEE Transactions on*, vol. 61, no. 1, pp. 266–278, 2013.

[11] F. Babich, M. Comisso, A. Crismani, and A. Dorni, "On the design of mac protocols for multi-packet communication in ieee 802.11 heterogeneous networks using adaptive antenna arrays," *Mobile Computing, IEEE Transactions on*, vol. 14, no. 11, pp. 2332–2348, 2015.

[12] K. C.-J. Lin, S. Gollakota, and D. Katabi, "Random access heterogeneous mimo networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 146–157, 2011.

[13] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless networks*, vol. 11, no. 4, pp. 471–487, 2005.

[14] D. Niculescu, "Interference map for 802.11 networks," in *Proceedings of ACM IMC*, 2007, pp. 339–350.

[15] N. Ahmed and S. Keshav, "Smarta: a self-managing architecture for thin access points," in *Proceedings of ACM CoNEXT*, 2006, p. 9.

[16] V. Shrivastava, S. Rayanchu, S. Banerjee, and K. Papagiannaki, "Pie in the sky: online passive interference estimation for enterprise wlans," in *Proceedings of USENIX NSDI*. USENIX Association, 2011, pp. 337–350.

[17] Y. C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage, "Jigsaw," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 39–50, 2006.

[18] J. Padhye, S. Agarwal, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill, "Estimation of link interference in static multi-hop wireless networks," in *Proceedings of ACM IMC*. USENIX Association, 2005, pp. 28–28.

[19] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 159–170, 2011.

[20] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11 b," in *Proceedings of IEEE INFOCOM*, vol. 2, 2003, pp. 836–843.

[21] QualNet, http://web.scalable-networks.com/.

[22] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Analyzing the mac-level behavior of wireless networks in the wild," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 75–86.

[23] N. Ahmed, U. Ismail, S. Keshav, and K. Papagiannaki, "Measuring multi-parameter conflict graphs for 802.11 networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 13, no. 3, pp. 54–57, 2010.

[24] J. Eriksson, S. Agarwal, P. Bahl, and J. Padhye, "Feasibility study of mesh networks for all-wireless offices," in *Proceedings of ACM MobiSys*, 2006, pp. 69–82.

[25] Iperf, http://iperf.fr.

[26] D. L. Mills, "Internet time synchronization: the network time protocol," *Communications, IEEE Transactions on*, vol. 39, no. 10, pp. 1482–1493, 1991.

[27] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.

On the Delay Performance of Interference Channels

Sebastian Schiessl[§], Farshad Naghibi[§], Hussein Al-Zubaidy[§], Markus Fidler[‡], James Gross[§]
[§]School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden
[‡]Institute of Communications Technology, Leibniz Universität Hannover, Germany

***Abstract*—A deep understanding of the queuing performance of wireless networks is essential for the advancement of future wireless communications. The stochastic nature of wireless channels in general gives rise to a time varying transmission rate. In such an environment, interference is increasingly becoming a key constraint. Obtaining an expressive model for offered service of such channels has major implications in the design and optimization of future networks. However, interference channels are not well-understood with respect to their higher layer performance. The particular difficulty for handling interference channels arises from the superposition of random fading processes for the signals of the transmitters involved (i.e., for the signal of interest and for the signals of the interferers). Starting from the distribution of the signal-to-interference-plus-noise ratio (SINR), we derive a statistical characterization of the underlying service process in terms of its Mellin transform. Then, we adapt a recent stochastic network calculus approach for fading channels to derive measures of the queuing performance of single- and multi-hop wireless interference networks. Special cases of our solution include noise-limited and interference-limited systems. A key finding of our analysis is that for a given average signal and average sum interference power, the performance of interfered systems not only depends on the relative strength of the sum interference with respect to the signal-of-interest power, but also on the interference structure (i.e., the number of interferers) as well as the absolute levels.**

I. Introduction

Over the last decade interference has become the key bottleneck for the further evolution of wireless networks. With the advent and proliferation of broadband wireless communication services, this interference limitation is apparent in multiple ways. For unlicensed bands, the interference limitation is due to the constantly increasing number of deployed wireless systems, running heterogeneous technologies and not undergoing a deployment planning. This has led to quite crowded frequency bands which are facing a significant coexistence problem [1]. On the other hand, the need for higher communication rates has forced cellular network providers to operate advanced packet-switched networks with a frequency reuse of one, i.e., potentially introducing a significant interference coupling between neighboring cells of the same system. Consequently, (mitigating) the impact of interference in wireless communication networks has become an intense area of research recently [2], [3]. Despite the large research interest with respect to interference channels on the physical layer, little is known with respect to the impact of interference regarding the higher layers. In particular, from a fundamental point of view only few attempts have been made to characterize the interference channel from a queuing-theoretic perspective. Hence, models to study the performance of wireless (interfered) store-and-forward networks are lacking.

For instance, a higher-layer queuing analysis of wireless networks under the impact of interference is presented in [4] for sensor networks. However, the considered interference model relates to the so-called protocol model, where nodes avoid interference for example based on CSMA/CA coordination. Furthermore, the authors studied only average performance metrics by mapping the transmission behavior to standard G/G/1 queuing models. A related analysis in the context of multi-hop networks is presented in [5]. Here, the authors also study the protocol interference model and analyze multi-hop packet transmissions based on the G/G/1 open queuing network model, providing average performance metrics like the delay. However, the more subtle effects of interference on the physical layer due to fading are not taken into account, while the analysis also falls short of providing a characterization of the end-to-end delay distribution. The concept of effective capacity can address these issues, in particular it provides bounds on the tail of the delay of a single-hop communication systems. However, typically interference is characterized in this context only as an additional constant contribution to the noise, ignoring the fading characteristic of interference [6], [7]. Finally, [8] analyzes the interference channel with respect to scheduling stability in larger ad-hoc networks under the well-known Lyapunov stability framework. While stability is an important aspect of queuing networks, further relevant metrics like the delay distribution cannot be addressed by this analysis.

In summary, most of the above works make a significant contribution towards understanding the average queuing performance of wireless networks under interference. However, they cannot provide a more fine-grained analysis especially of the delay distributions. Furthermore, a common assumption among these works is the transmission of a single constant rate traffic stream over a single interference channel. The characterization of multi-hop performance of interference channels for variable rate traffic streams, in particular with respect to delay distributions, is an open challenge. This is especially true when it comes to precise models of the physical layer that also take into account the fading of the interference signals. This clearly limits these approaches in terms of their expressibility with respect to fading profiles or transmit power settings.

In this work, we provide a network-layer performance analysis of interference channels in terms of their fading parameters. To our knowledge, this aspect has not been addressed

This work was supported in part by the European Research Council (ERC) under Starting Grant UnIQue (StG 306644).

ISBN 978-3-901882-83-8 © 2016 IFIP

before. To enable such analysis, we utilize recent results from the literature, namely (i) the fading distribution of interference channels [9], and (ii) the $(\min, \times)$ network calculus for wireless network analysis [10]. A key step in such analysis is the characterization of the service element (in this case the interference channel) which then enables the determination of the desired performance bounds. Our main contribution is to provide such mathematical characterization of the service offered by the interference channel, i.e., a fading channel in the presence of multiple (fading) interferers, in terms of the Mellin transform of the cumulative service process of the channel. Computing this particular Mellin transform involves the solution of an integral transform of a *ratio distribution* which is known to be notoriously difficult to handle. We then use the resulting service process to obtain probabilistic bounds on the delay performance and provide the corresponding delay performance for some special cases.

This contribution has four main and novel implications: First, our analysis revealed that when the interference power is time-varying due to channel fading, then for a given total average interference power, the performance of interfered wireless systems depends heavily on the structure of the interference rather than just the average interference power. In particular, characterizing the interference as constant noise leads to wrong performance assumptions of the system. Second, the mathematical treatment of wireless systems with interference channels that we propose here can serve as the basis for a system-level, cross-layer optimization. Third, due to the network calculus approach used in this work, the obtained results can be easily extended to multi-hop settings, as we show in Section III-C. Finally, the capacity expression for many other interesting channels, e.g., the secrecy channel, have similar structure to that of the interference channel. Hence, one can use our proposed approach and results to investigate the performance of such channels.

The rest of the paper is structured as follows: In Section II, we provide the required background on the stochastic network calculus. The network calculus model of the interference channel is derived in Section III. Our numerical investigations are presented in Section IV. Finally, we provide brief conclusions in Section V.

II. Stochastic Network Calculus for Wireless Channels

In this section, we provide a brief description of the stochastic network calculus and its application to fading channels. We then adapt this network calculus to interference channels in the following section. The reader may refer to [10]–[18] for more details on the network calculus and to [19]–[23] for applications to wireless, fading, and Gilbert-Elliott channels.

Stochastic network calculus considers queuing systems and networks of systems with stochastic arrival, departure, and service processes, where the bivariate functions $A(\tau,t)$, $D(\tau,t)$ and $S(\tau,t)$ for any $0 \leq \tau \leq t$ denote the *cumulative* arrivals to the system, departures from the system, and service offered by the system, respectively, in the interval $[\tau, t)$. We consider a discrete time model, where time-slots have a duration T and $t \geq 0$ denotes the index of the respective time-slot.

A lossless system with service process $S(\tau,t)$ satisfies the input/output relationship $D(0,t) \geq A \otimes S\,(0,t)$, where $\otimes$ is the $(\min,+)$ convolution operator given by $x \otimes y\,(\tau,t) = \inf_{\tau \leq u \leq t} \{x(\tau,u) + y(u,t)\}$. A network service process of an H-hop path can be obtained using the $(\min,+)$ convolution, i.e., $S_{\text{net}} = S_1 \otimes S_2 \otimes \cdots \otimes S_H$. In general, we are interested in probabilistic bounds of the form $\Pr[W(t) > w^{\varepsilon}] \leq \varepsilon$, which is also known as the *violation probability* for a target delay w^{ε}, under stable system operation.

Modeling wireless links in the context of network calculus however is not a trivial task. A particular difficulty arises when we seek to obtain a stochastic characterization of the cumulative service process of a wireless fading channel, as also witnessed in the context of the effective capacity of wireless systems [24]. A promising, recent approach for wireless networks has been proposed in [10] where the queuing behavior is analyzed directly in the "domain" of channel variations instead of the bit domain [10], [25]. This can be interpreted as the *SNR domain* (thinking of bits as "SNR demands" that reside in the system until these demands can be met by the channel).

To start with, the cumulative arrival, departure, and service processes in the bit domain, i.e., A, D, and S, are related to their SNR domain counterparts (represented in the following by calligraphic capital letters $\mathcal{A}$, $\mathcal{D}$, and $\mathcal{S}$) respectively, through the exponential function. Thus, we have $\mathcal{A}(\tau,t) \triangleq e^{A(\tau,t)}$, $\mathcal{D}(\tau,t) \triangleq e^{D(\tau,t)}$, and $\mathcal{S}(\tau,t) \triangleq e^{S(\tau,t)}$. Due to the exponential function, these cumulative functions become products of the increments in the bit domain. Assuming Shannon capacity

$$c_t = \log g\,(\gamma_t) = N \log_2 (1+\gamma_t)\,, \tag{1}$$

where c_t is the random service offered by the system in time-slot t, N is the number of transmitted symbols per time-slot, and γ_t is the instantaneous SNR, we obtain the cumulative service process in the SNR domain as

$$\mathcal{S}(\tau,t) = \prod_{u=\tau}^{t-1} e^{c_u} = \prod_{u=\tau}^{t-1} g\,(\gamma_u) = \prod_{u=\tau}^{t-1} (1+\gamma_u)^{\mathcal{N}}\,, \tag{2}$$

where $\mathcal{N} = N/\log 2$. To simplify notation, we consider the case $\mathcal{N} = 1$ in the following. Performance bounds for the general case can be obtained by appropriately scaling the obtained results. Furthermore, in case of first-come first-served order, the delay at time t is obtained as follows

$$W(t) = \mathcal{W}(t) = \inf\{u \geq 0 : \mathcal{A}(0,t)/\mathcal{D}(0,t+u) \leq 1\}. \tag{3}$$

A bound ε for the delay violation probability $\Pr[W(t) > w^{\varepsilon}]$ can be derived based on a transform of the cumulative arrival and service processes in the SNR domain using the moment bound. In [10], it was shown that such a violation probability bound for a given w^{ε} can be obtained as

$$\varepsilon = \inf_{s>0} \{\mathcal{K}(s, t+w^{\varepsilon}, t)\}\,. \tag{4}$$

We refer to the function $\mathcal{K}(s,\tau,t)$ as the *kernel* defined as

$$\mathcal{K}(s,\tau,t) = \sum_{u=0}^{\min(\tau,t)} \mathcal{M}_{\mathcal{A}}(1+s,u,t)\mathcal{M}_{\mathcal{S}}(1-s,u,\tau), \quad (5)$$

where the function $\mathcal{M}_{\mathcal{X}}(s)$ is the Mellin transform [26] of a random process, defined as

$$\mathcal{M}_{\mathcal{X}}(s,\tau,t) = \mathcal{M}_{\mathcal{X}(\tau,t)}(s) = \mathrm{E}\left[\mathcal{X}^{s-1}(\tau,t)\right], \quad (6)$$

for any $s \in \mathbb{C}$, where we restrict our derivations in this work to real valued $s \in \mathbb{R}$. We note that by definition of $\mathcal{X}(\tau,t) = e^{X(\tau,t)}$, the Mellin transform $\mathcal{M}_{\mathcal{X}}(s,\tau,t) = \mathrm{E}\left[e^{(s-1)X(\tau,t)}\right]$ after substitution of parameter $s = \theta+1$ implies also a solution for the moment-generating function (MGF), that is the basis of the effective capacity model [24] and of an MGF network calculus [15].

In the following, we will assume $\mathcal{A}(\tau,t)$ and $\mathcal{S}(\tau,t)$ to have stationary and independent increments. We denote them by α for the arrivals (in SNR domain) and $g(\gamma)$ for the service. Hence, the Mellin transforms become independent of the time instance, which we account for by denoting $\mathcal{M}_{\mathcal{X}}(s,\tau,t) = \mathcal{M}_{\mathcal{X}}(s,t-\tau)$. In addition, as we only consider stable queuing systems in steady-state, the kernel becomes independent of the time instance t and we denote $\mathcal{K}(s,t+w,t) \overset{t\to\infty}{=} \mathcal{K}(s,-w)$.

The strength of the Mellin-transform-based approach becomes apparent when considering block-fading channels. The Mellin transform for the cumulative service process in SNR domain is given by

$$\mathcal{M}_{\mathcal{S}}(s,\tau,t) = \prod_{u=\tau}^{t-1} \mathcal{M}_{g(\gamma)}(s) = \mathcal{M}_{g(\gamma)}^{t-\tau}(s) = \mathcal{M}_{\mathcal{S}}(s,t-\tau),$$

where $\mathcal{M}_{g(\gamma)}(s)$ is the Mellin transform of the stationary and independent service increment $g(\gamma)$ in the SNR domain. The function $g(\cdot)$ represents here the modification of the SNR due to the Shannon formula Eq. (1). However, it can also model more complex system characteristics, most importantly scheduling effects.

Assuming the cumulative arrival process in SNR domain to have stationary and independent increments and denoting the corresponding Mellin transform by $\mathcal{M}_{\mathcal{A}}(s,t-\tau) = \prod_{i=\tau}^{t-1} \mathcal{M}_{\alpha}(s)$, the steady-state kernel for a fading wireless channel is given by [10]

$$\mathcal{K}(s,-w) = \frac{\mathcal{M}_{g(\gamma)}^{w}(1-s)}{1-\mathcal{M}_{\alpha}(1+s)\,\mathcal{M}_{g(\gamma)}(1-s)} \quad (7)$$

for any $s > 0$, under the stability condition

$$\mathcal{M}_{\alpha}(1+s)\,\mathcal{M}_{g(\gamma)}(1-s) < 1. \quad (8)$$

Assuming Rayleigh fading, i.e., an exponentially distributed SNR with average $\gamma_{\emptyset}$ at the receiver, the Mellin transform results into [10]

$$\mathcal{M}_{g(\gamma)}(s) = e^{\frac{1}{\gamma_{\emptyset}}}{\gamma_{\emptyset}}^{s-1}\Gamma\left(s,\gamma_{\emptyset}^{-1}\right). \quad (9)$$

where $\Gamma(x,y) = \int_y^{\infty} t^{x-1}e^{-t}\mathrm{d}t$ is the incomplete Gamma function. Then the steady-state kernel for a Rayleigh-fading wireless channel turns out to be

$$\mathcal{K}(s,-w) = \frac{\left(e^{1/\gamma_{\emptyset}}{\gamma_{\emptyset}}^{-s}\Gamma\left(1-s,\frac{1}{\gamma_{\emptyset}}\right)\right)^{w}}{1-\mathcal{M}_{\alpha}(1+s)\,e^{1/\gamma_{\emptyset}}{\gamma_{\emptyset}}^{-s}\Gamma\left(1-s,\frac{1}{\gamma_{\emptyset}}\right)} \quad (10)$$

for any $s > 0$ and under the stability condition in Eq. (8). By substitution of the kernel Eq. (10) into Eq. (4), a bound of the delay violation probability ε for a given w^{ε} can be obtained.

III. Performance of Interference Channels

We first introduce a characterization of interference channels assuming independent block-fading processes for a transmitter/receiver pair and an arbitrary number of interferers. We then use this channel model to compute probabilistic performance bounds for interference channels in terms of their fading parameters.

A. Block-Fading Interference Channel

Consider a wireless communication scenario with one transmitter/receiver pair that is subject to interfering signals from a set $\mathbf{I}$ of interferers. Index $i \leq |\mathbf{I}|$ denotes the link between interferer i and the receiver, while $i = 0$ denotes the link between the transmitter and the receiver (i.e., the signal of interest). Denote by P_i the transmit power per link, i.e., P_0 denotes the transmit power of the signal of interest, P_1 denotes the transmit power of interferer 1, and so on.

The received power varies from time slot to time slot due to randomly varying channel gains of all links. Denote the random channel gain of link i during slot t by $h_{i,t}$. We focus in the following on random variations due to independent Rayleigh-distributed block-fading processes for all links i. Hence, the received signal strength of link i is given by $P_i|h_{i,t}|^2$ and is exponentially distributed with mean $p_i = P_i \cdot \mathrm{E}\left[|h_i|^2\right]$. The fading is assumed to stay constant during one slot but varies independently from slot to slot. Furthermore, the fading between different links is assumed to be statistically independent. Based on the above definitions, the instantaneous signal-to-interference-plus-noise ratio (SINR) at the receiver during slot t is a random variable and given as

$$\gamma_t = \frac{P_0|h_{0,t}|^2}{\sum_i P_i|h_{i,t}|^2 + \sigma^2}, \quad (11)$$

where σ^2 denotes the power of the additive white Gaussian noise (AWGN) process at the receiver. Depending on the SINR, which is assumed to be known at the transmitter, the amount of information that can be conveyed changes in each time slot. In this work, we consider that for an SINR γ_t the transmitter is able to transmit c_t bits correctly to the receiver during slot t, where c_t is defined by Eq. (1).

B. Derivation of $\mathcal{M}_{g(\gamma)}(s)$

Based on the system model in Section III-A, we proceed to present the main contributions of the paper. Initially, we concentrate on deriving the Mellin transform of the service process for the interference channel. Then, we use the result to compute the kernel in Eq. (7). Recall that we assume, apart from the signal of interest, $|\mathbf{I}|$ interferers to be present. The

resulting SINR γ_t is given by the ratio of exponentially distributed random variables in Eq. (11). Considering stationary and independent γ_t for all t, we omit the index t. For γ we have the distribution function [27]

$$\Pr[\gamma \leq x] = F_\gamma(x) = 1 - e^{\frac{-x}{\gamma_\emptyset}} \prod_{\forall i \in \mathbf{I}} \frac{a_i}{a_i + x}, \quad (12)$$

where $a_i = \frac{p_0}{p_i}$ denotes the ratio of the average received power from the signal of interest and interferer i. Furthermore, $\gamma_\emptyset = \frac{p_0}{\sigma^2}$ is the noise-limited average SNR of the signal of interest. Based on partial fractions decomposition [28], it was shown [9] that the CDF in Eq. (12) can be reorganized as

$$F_\gamma(x) = 1 - e^{\frac{-x}{\gamma_\emptyset}} \sum_{\forall i \in \mathbf{I}} \frac{u_i}{a_i + x}, \quad (13)$$

where $u_i = \prod_{\forall s \in \mathbf{I}} a_s \left(\prod_{\forall t \in \mathbf{I} \setminus \{i\}} (a_t - a_i) \right)^{-1}$ if there are multiple interferers $|\mathbf{I}| \geq 2$, and $u_1 = a_1$ if there is only one interferer $|\mathbf{I}| = 1$. Note that the above representation only holds for interferers with different interference strengths, that is $a_i \neq a_j$ for all $i \neq j$.

First, we determine the Mellin transform for $g(\gamma)$, where γ is the instantaneous SINR distributed according to Eq. (13). The Mellin transform is given by

$$\mathcal{M}_{g(\gamma)}(s) = \mathrm{E}\left[g(\gamma)^{s-1}\right] = \int_0^\infty (1+x)^{s-1} \, \mathrm{d}F_\gamma(x) \quad (14)$$

for $s < 1$. To solve the integration above, we need the following lemma.

Lemma 1.

$$\int_0^\infty \frac{(1+x)^{s-2}}{a_i + x} e^{\frac{-x}{\gamma_\emptyset}} \mathrm{d}x = e^{\frac{1}{\gamma_\emptyset}} \left(I_1^\infty(s) + I_\delta(s) + I_2^\infty(s) \right),$$

for any small $\delta > 0$, where

$$I_1^k(s) = \sum_{n=0}^{k} \frac{(-1)^n \gamma_\emptyset^{s+n-1}}{(a_i - 1)^{n+1}} \left[\Gamma\left(s+n-1, \frac{1}{\gamma_\emptyset}\right) - \Gamma\left(s+n-1, \frac{a_i - 1 - \delta)}{\gamma_\emptyset}\right) \right], \quad (15)$$

for $a_i > 2 + \delta$ and $I_1^k(s) = 0$, otherwise;

$$I_\delta(s) = \int_{\max(1, a_i - 1 - \delta)}^{\max(1, a_i - 1 + \delta)} \frac{z^{s-2}}{z + a_i - 1} e^{\frac{-z}{\gamma_\emptyset}} \mathrm{d}z \quad (16)$$

and

$$I_2^k(s) = \sum_{n=0}^{k} (1 - a_i)^n \gamma_\emptyset^{s-n-2} \Gamma\left(s - n - 2, \frac{\max(1, a_i - 1 + \delta)}{\gamma_\emptyset}\right) \quad (17)$$

for $k \geq 0$.

Proof: To solve the integral in Lemma 1, we start by performing a change of variable and letting $z = x + 1$. Then,

$$\int_0^\infty \frac{(1+x)^{s-2}}{x + a_i} e^{\frac{-x}{\gamma_\emptyset}} \mathrm{d}x = e^{\frac{1}{\gamma_\emptyset}} \int_1^\infty \frac{z^{s-2}}{z + a_i - 1} e^{\frac{-z}{\gamma_\emptyset}} \mathrm{d}z. \quad (18)$$

Note that since $a_i > 0$ by definition and $z \in [1, \infty)$, we have $z + a_i - 1 > 0$.

For the integral in the right hand side of Eq. (18), we use the following series representation

$$\frac{1}{b+1} = \sum_{n=0}^{\infty} (-1)^n b^n, \quad \text{for } |b| < 1. \quad (19)$$

To ensure that $|b| < 1$, we partition the integral in Eq. (18) into three parts given as

$$\int_1^\infty f(z)\mathrm{d}z = \int_1^{\max(1, a_i - 1 - \delta)} f(z)\mathrm{d}z + \int_{\max(1, a_i - 1 - \delta)}^{\max(1, a_i - 1 + \delta)} f(z)\mathrm{d}z + \int_{\max(1, a_i - 1 + \delta)}^\infty f(z)\mathrm{d}z. \quad (20)$$

In the expression above, the second term, which is Eq. (16), diminishes when $\delta \to 0$. Depending on the range of values of a_i, we identify three cases: (i) $a_i \in (0, 2-\delta]$, then the first term and the second term will evaluate to zero and the integral in the third term will start at one; (ii) $a_i \in (2-\delta, 2+\delta]$, then the first term will again evaluate to zero, the integral in the second term will start at one, while the integral in the third term will start at $a_i - 1 + \delta$; and the last case (iii) $a_i \in (2+\delta, \infty)$, then the first term will have a value greater than zero, the integral in the second term will start at $z = a_i - 1 - \delta$ and the integral in the third term will start at $z = a_i - 1 + \delta$.

We can now apply the series expansion to the first and third term of Eq. (20) above. Starting with the integral in the first term, after multiplication with $a_i - 1$ and considering the non-trivial case $a_i > 2 + \delta$, we have

$$\begin{aligned} &\int_1^{a_i - 1 - \delta} \frac{z^{s-2}}{\frac{z}{a_i - 1} + 1} e^{-\frac{z}{\gamma_\emptyset}} \mathrm{d}z = \int_1^{a_i - 1 - \delta} \sum_{n=0}^{\infty} \left(\frac{-z}{a_i - 1} \right)^n z^{s-2} e^{-\frac{z}{\gamma_\emptyset}} \mathrm{d}z \\ &= \sum_{n=0}^{\infty} \frac{(-1)^n \gamma_\emptyset^{s+n-1}}{(a_i - 1)^n} \int_{1/\gamma_\emptyset}^{(a_i - 1 - \delta)/\gamma_\emptyset} y^{s+n-2} e^{-y} \mathrm{d}y \\ &= (a_i - 1) I_1^\infty(s), \end{aligned} \quad (21)$$

where in the second line we used the change of variables $y = \frac{z}{\gamma_\emptyset}$. The last equality follows from the definition of the incomplete Gamma function.

For the third term in Eq. (20), we compute

$$\begin{aligned} &\int_{\max(1, a_i - 1 + \delta)}^\infty \frac{z^{s-3}}{1 + \frac{a_i - 1}{z}} e^{-\frac{z}{\gamma_\emptyset}} \mathrm{d}z \\ &= \int_{\max(1, a_i - 1 + \delta)}^\infty \sum_{n=0}^{\infty} \left(\frac{1 - a_i}{z} \right)^n z^{s-3} e^{-\frac{z}{\gamma_\emptyset}} \mathrm{d}z \end{aligned}$$

$$= \sum_{n=0}^{k} (1-a_i)^n \gamma_\emptyset^{s-n-2} \int_{\max(1,a_i-1+\delta)/\gamma_\emptyset}^{\infty} y^{s-n-3} e^{-y} \mathrm{d}y$$
$$= I_2^{\infty}(s). \quad (22)$$

Again we used the change of variables $y = \frac{z}{\gamma_\emptyset}$ in the third line and the definition of the incomplete Gamma function in the last line. The lemma follows by substituting Eq. (21) and Eq. (22) in Eq. (20) and by considering the range of a_i. ■

In Lemma 1, we decompose the integral into three terms in order to be able to handle the evaluation of the otherwise intractable integral. The second term ($I_\delta(s)$) diminishes when δ approaches 0 and may be ignored for $\delta \ll 1$. The first term ($I_1(s)$) has a value only when $a_i > 2$, i.e., when $p_0 > 2p_i$ and is zero otherwise. The third term contributes to the solution for the entire range of a_i and represents the complete solution when $a_i < 2$, i.e., $p_0 < 2p_i$.

Theorem 1. *For the interference channel, we have*

$$\mathcal{M}_{g(\gamma)}(s) = 1 + \sum_{\forall i \in \mathbf{I}} u_i \left((s-1) e^{\frac{1}{\gamma_\emptyset}} \left(I_1^{\infty}(s) + I_\delta(s) + I_2^{\infty}(s) \right) \right),$$

for any $s < 1$.

Proof: The Mellin transform of $g(\gamma)$ is given by Eq. (14) as

$$\mathcal{M}_{g(\gamma)}(s) = \int_0^{\infty} (1+x)^{s-1} \, \mathrm{d}F_\gamma(x). \quad (23)$$

Using integration by parts, we obtain for $s < 1$ that

$$\begin{aligned}
\mathcal{M}_{g(\gamma)}(s) =& (1+x)^{s-1} F_\gamma(x) \Big|_0^{\infty} \\
& - (s-1) \int_0^{\infty} (1+x)^{s-2} F_\gamma(x) \, \mathrm{d}x \\
=& - (s-1) \int_0^{\infty} (1+x)^{s-2} \mathrm{d}x \\
& + (s-1) \sum_{\forall i \in \mathbf{I}} u_i \int_0^{\infty} \frac{(1+x)^{s-2}}{a_i + x} e^{\frac{-x}{\gamma_\emptyset}} \, \mathrm{d}x \\
=& 1 + (s-1) \sum_{\forall i \in \mathbf{I}} u_i \int_0^{\infty} \frac{(1+x)^{s-2}}{a_i + x} e^{\frac{-x}{\gamma_\emptyset}} \, \mathrm{d}x,
\end{aligned}$$

where in the second step we inserted Eq. (13). ■

Lemma 1 gives the exact solution of the individual terms of Theorem 1 expressed by infinite sums of incomplete Gamma functions. For numerical evaluation this may pose a computational problem. The following corollary provides easily computable bounds through truncation of these sums.

Corollary 1. *For any even $k > 0$, $a_i > 1$, and $s < 1$,*

$$\Psi_i^{k+1}(s) \leq \int_0^{\infty} \frac{(1+x)^{s-2}}{a_i + x} e^{\frac{-x}{\gamma_\emptyset}} \, \mathrm{d}x \leq \Psi_i^k(s), \quad (24)$$

where

$$\Psi_i^k(s) = \lim_{\delta \to 0} e^{\frac{1}{\gamma_\emptyset}} \left(I_1^k(s) + I_\delta(s) + I_2^k(s) \right).$$

The approximation error is bounded by $|\Psi^{k+1} - \Psi^k|$.

Proof: The proof follows directly from the monotonicity of the series expansion in Eq. (19) that has a limit of zero and the Leibniz alternating series test. The approximation error of the partial sum for $n = 0, \ldots, k$, where k is an even integer, is upper bounded by the $(k+1)^{\text{th}}$ element of the series. ■

The corollary above provides a practical way to bound the integral in Lemma 1. In general, we are interested in an upper bound on the Mellin transform of the service $\mathcal{M}_{g(\gamma)}(s)$ for $s < 1$ which provides an upper bound on the delay violation probability. Thus, truncation of the series at an even k always leads to valid delay bounds. The truncation error can be made arbitrarily small by choosing larger k. For the case $a_i < 1$ the series expansion of Lemma 1 uses the geometric series, where a truncation provides an approximate solution for the Mellin transform of the service. Again due to the convergence of the series, the truncation error can be made arbitrarily small by the choice of k.

Network layer performance bounds, e.g., a probabilistic delay bound, for a network of nodes with interference channels can be readily obtained from the Mellin transform of the service process of the channel which is characterized by Theorem 1 and results from the network calculus presented in Section II. The numerical results are presented in Section IV.

C. Asymptotes, Special Cases, and Multi-Hop Networks

In this subsection, we consider special cases, such as the noise-limited and the interference-limited channels. Also, we provide a solution of multi-hop networks.

a) Noise-Limited Channel: In this case $p_0 \gg p_i$ so that $a_i = p_0/p_i \to \infty$. It follows that Eq. (12) evaluates to

$$F_\gamma(x) = 1 - e^{\frac{-x}{\gamma_\emptyset}},$$

which is the CDF of the Rayleigh fading channel.

Considering Theorem 1 and the case of a single interferer with $u_i = a_i$ the Mellin transform evaluates to

$$\mathcal{M}_{g(\gamma)}(s) = \left(1 + (s-1) e^{\frac{1}{\gamma_\emptyset}} a_i I_1^{\infty}(s) \right),$$

where we used that Eq. (16) and Eq. (22) tend to zero for $a_i \to \infty$. The term $a_i I_1^{\infty}(s)$, where $I_1^{\infty}(s)$ is given by Eq. (15), evaluates for $a_i \to \infty$ to

$$a_i I_1^{\infty}(s) = \gamma_\emptyset^{s-1} \Gamma\left(s-1, \frac{1}{\gamma_\emptyset} \right),$$

where we used that only the summand at $n = 0$ does not tend to zero. The Mellin transform follows as

$$\mathcal{M}_{g(\gamma)}(s) = 1 + (s-1) e^{\frac{1}{\gamma_\emptyset}} \gamma_\emptyset^{s-1} \Gamma\left(s-1, \frac{1}{\gamma_\emptyset} \right). \quad (25)$$

Using the recurrence relation $\Gamma(s,x) = (s-1)\Gamma(s-1,x) + x^{s-1}e^{-x}$ we find that Eq. (25) is equivalent to the Mellin transform of the Rayleigh fading channel that was previously found in [10], i.e., Eq. (9) is recovered.

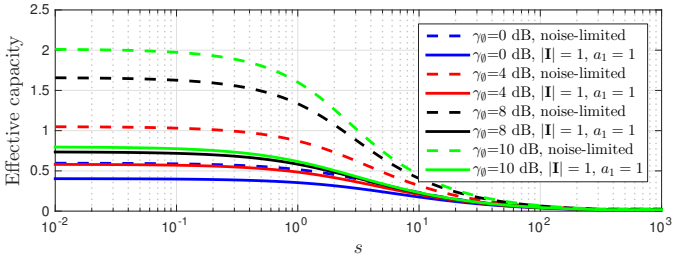


Fig. 1. Effective capacity (defined as $\frac{-1}{s}\log \mathcal{M}_{g(\gamma)}(1-s)$) as a function of s for $\gamma_\emptyset \in \{0, 4, 8, 10\}$ dB, for an interference channel with a single interferer and identical average received power compared to noise-limited channel, assuming Rayleigh fading.

b) Large Number of Interferers: For a large number of interferers $|\mathbf{I}|$ with independent fading, the central limit theorem predicts that the combined interference power at the receiver becomes a Gaussian random variable. Consequently, when the total power of the interference is split between infinitely many interferers, the variance of that random variable goes to zero and the total interference power at the receiver will have constant power $p_\mathbf{I}$. This has the same effect as an additional noise term with constant power $p_\mathbf{I}$, and thus the channel will behave like a noise-limited channel whose average SNR is equal to the average SINR $\frac{p_0}{\sigma^2+p_\mathbf{I}}$ of the interference channel.

c) Identical Average Received Power: We consider the special case where the average received power of the signal of interest and the signal of interferer i are identical, i.e., $p_0 = p_i$ so that parameter a_i evaluates to $a_i = 1$. By insertion of $a_i = 1$ into Lemma 1, we obtain $I_1^\infty(s) = 0$. Letting $\delta \to 0$, it follows that $I_\delta(s) \to 0$ diminishes. Finally, the first factor of I_2^∞ given by Eq. (17), i.e., $(1-a_i)^n$, is defined to be one for $n = 0$ and zero otherwise, so that Lemma 1 evaluates to the simpler form

$$\int_0^\infty \frac{(1+x)^{s-2}}{a_i+x} e^{\frac{-x}{\gamma_\emptyset}}\,\mathrm{d}x = e^{\frac{1}{\gamma_\emptyset}}\gamma_\emptyset^{s-2}\Gamma\left(s-2, \frac{1}{\gamma_\emptyset}\right).$$

For this special case, the above integral can also be solved directly without using the series expansion of Lemma 1, resulting in the same solution.

Considering a single interferer, the Mellin transform from Theorem 1 becomes

$$\mathcal{M}_{g(\gamma)}(s) = 1 + (s-1)e^{\frac{1}{\gamma_\emptyset}}\gamma_\emptyset^{s-2}\Gamma\left(s-2, \frac{1}{\gamma_\emptyset}\right), \quad (26)$$

which closely resembles the form of the Rayleigh fading channel in Eq. (25).

Fig. 1 depicts the effective capacity, computed as the normalized log Mellin transform of the service process defined as $\frac{-1}{s}\log \mathcal{M}_{g(\gamma)}(1-s)$, against parameter $s > 0$ for the special case (c) and compared to the noise-limited channel in case (a). As expected, when s approaches zero, the effective capacity approaches the average channel capacity. As we increase s (i.e., when a flow demands better QoS than mere average guarantees), the effective capacity diminishes and approaches the minimal capacity, that is zero. As expressed by Eqs. (25) and (26), the general shape of the effective capacity is the same for the case with and without interference. While the effective capacity of the noise-limited channel improves significantly as the channel SNR (i.e., $\gamma_\emptyset$) increases, the improvement is much smaller for the case with interference, and the effect of interference becomes more prominent with increasing SNR.

d) Interference-Limited Channel: We characterize the interference-limited case by considering $\gamma_\emptyset \to \infty$, i.e., the noise power diminishes against the signal of interest as well as the interfering signals. Consequently, the distribution in Eq. (13) reduces to

$$F_\gamma(x) = 1 - \sum_{\forall i \in \mathbf{I}} \frac{u_i}{a_i + x}. \quad (27)$$

This leads to a structurally simpler solution. The exponential term in the integral of Lemma 1 disappears. The following lemma provides the solution for the new integral.

Lemma 2.

$$\int_0^\infty \frac{(1+x)^{s-2}}{a_i+x}\mathrm{d}x = \hat{I}_1^\infty(s) + \hat{I}_\delta(s) + \hat{I}_2^\infty(s),$$

for any small $\delta > 0$, where

$$\hat{I}_1^k(s) = \sum_{n=0}^{k} \frac{(-1)^n}{(a_i-1)^n}\left(\frac{(a_i-1-\delta)^{s-1+n}-1}{s-1+n}\right), \quad (28)$$

for $a_i > 2+\delta$ and $\hat{I}_1^k(s) = 0$, otherwise;

$$\hat{I}_\delta(s) = \int_{\max(1,a_i-1-\delta)}^{\max(1,a_i-1+\delta)} \frac{z^{s-2}}{z+a_i-1}\mathrm{d}z \quad (29)$$

and

$$\hat{I}_2^k(s) = \sum_{n=0}^{k} (1-a_i)^n \frac{(a_i-1+\delta)^{s-2-n}}{s-2-n} \quad (30)$$

for $k \geq 0$.

The proof follows very closely the one of Lemma 1 except that we consider Eq. (21) and Eq. (22) with respect to the new distribution given in Eq. (27). We omit the detailed proof due to space constraints. The corresponding Mellin transform can then be obtained by applying Theorem 1 using Lemma 2 instead of Lemma 1.

e) Multi-Hop Interference Networks: Stochastic network calculus allows the representation of a multi-hop network by a single network service process $\mathcal{S}_{\text{net}}$, which is obtained by concatenating the service processes for all H nodes along the traversed path, i.e.,

$$\mathcal{S}_{\text{net}}(\tau, t) = \mathcal{S}_1 \otimes \mathcal{S}_2 \otimes \cdots \otimes \mathcal{S}_H(\tau, t), \quad (31)$$

where $\otimes$ is the $(\min, \times)$ convolution operator defined in Section II.

A bound on the convolution of two independent service processes $\mathcal{S}_1(\tau,t)$ and $\mathcal{S}_2(\tau,t)$ can be obtained using the Mellin transform, for any $s<1$, as

$$\mathcal{M}_{\mathcal{S}_1\otimes\mathcal{S}_2}(s,\tau,t)\leq\sum_{u=\tau}^{t}\mathcal{M}_{\mathcal{S}_1}(s,\tau,u)\mathcal{M}_{\mathcal{S}_2}(s,u,t). \quad (32)$$

For a cascade of H independent and identically distributed (i.i.d.) fading channels, we obtain for $s<1$ [25],

$$\mathcal{M}_{\mathcal{S}_{\text{net}}}(s,\tau,t)\leq\binom{H-1+t-\tau}{t-\tau}\left(\mathcal{M}_{g(\gamma)}(s)\right)^{t-\tau}, \quad (33)$$

where $\mathcal{M}_{g(\gamma)}(s)$ is given by Theorem 1. When $a_i>1$ and to simplify the computation of the desired performance bound, we use Corollary 1 to bound $\mathcal{M}_{g(\gamma)}(s)$, then Eq. (33) becomes

$$\mathcal{M}_{\mathcal{S}_{\text{net}}}(s,\tau,t)\leq\binom{H-1+t-\tau}{t-\tau}\cdot\left(1+\sum_{\forall i\in\mathbf{I}}u_i\left((s-1)\Psi_i^k(s)\right)\right)^{t-\tau},$$

for any $s<1$. Substituting the above in Eq. (5) gives the desired network performance bound. For a cascade of channels that are independent but have different distributions, the joint Mellin transform can still be obtained from the individual ones, however the expressions are more complex [29].

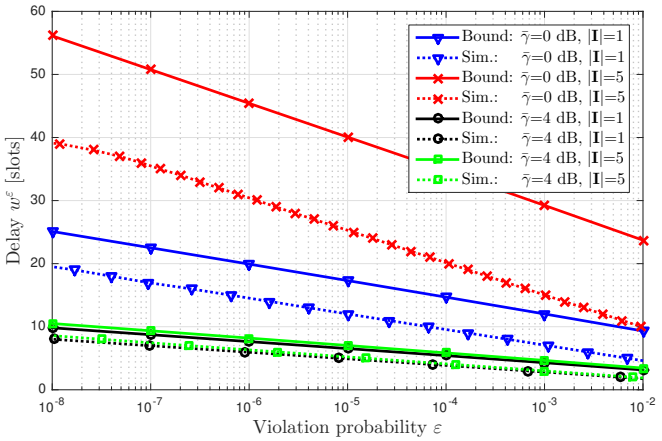
IV. Numerical Investigation

In this section, we conduct numerical investigations of the interference channel performance based on the analysis presented in the previous section. We first validate our analytical results using simulations. Then we use the analytical model to address several important questions regarding the structure of the interference channel and its impact on the system performance. In particular, we focus on the number of interferers as well as their absolute strengths, i.e., their average powers. We define the ratio of average received signal power to average interference-plus-noise power at the receiver as $\overline{\gamma}=\frac{p_0}{\sigma^2+\sum_i p_i}$, which is equivalent to the average SINR of a system where the total interference power is considered as an additive contribution to the noise.

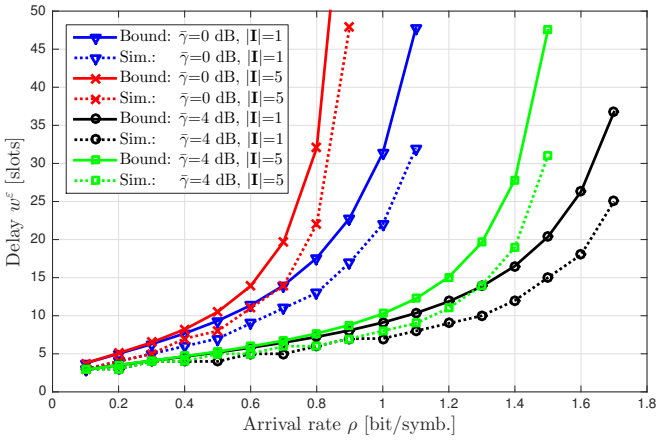
In all of the following investigations, we choose the arrival model to be a constant rate process with rate ρ measured in bits per time slot. The Mellin transform of the arrival process is then given by $\mathcal{M}_{\mathcal{A}}(s,t-\tau)=e^{\rho(s-1)(t-\tau)}$. Using this Mellin transform and that of the service process derived in Section III for the interference network, we obtain the kernel in Eq. (7) and consequently the bound on the delay and violation probability based on Eq. (4).

A. Validation of the Analytical Bounds

We validate our computed bounds using simulation. We simulate a queuing system with service process given by the channel capacity of the interference channel, an arrival process with constant rate ρ and with FIFO service discipline. In order to estimate the target delay violation probabilities in the order of 10^{-6}, we run the simulation for 10^{10} slots. In Fig. 2a we show the delay bound (w^ε) measured in transmission slots versus the violation probability ($\Pr[W(t)>w^\varepsilon]\leq\varepsilon$) and compare it to the simulated delay. We study the delay performance for different combinations of $\overline{\gamma}$ and number of interferers ($|\mathbf{I}|$), while we keep the arrival rate and the average SNR $\gamma_\emptyset=15$ dB constant for all curves. These combinations reflect a wide delay performance range. As expected, we observe that the analytical results (solid curves) indeed are upper bounds for the performance of their corresponding simulated systems (dotted curves). Furthermore, we observe that the bounds are tight enough for a reasonable estimation of the system performance. It also shows that bounds grow tighter as the system becomes less utilized. More importantly, in all cases the slope (i.e., the exponential decay) of the analytical and simulated curves match, therefore the relative gap diminishes as the delay grows larger.



(a)



(b)

Fig. 2. Validation of the analysis and delay performance with simulations for the parameter $\bar{\gamma}\in\{0,4\}$ and number of interferers $|\mathbf{I}|\in\{1,5\}$ with fixed average SNR $\gamma_\emptyset=15$ dB: (a) Delay bound (w^ε) in slots versus violation probability (ε) with fixed arrival rate $\rho=0.85$ bit/symbol. (b) Delay bound (w^ε) in slots versus arrival rate (ρ) in bit/symbol with fixed $\varepsilon=10^{-6}$.

In Fig. 2b, we show the delay bound (w^ε) measured in transmission slots versus the arrival rate for a violation probability of $\varepsilon=10^{-6}$ and using the same parameter combinations as in

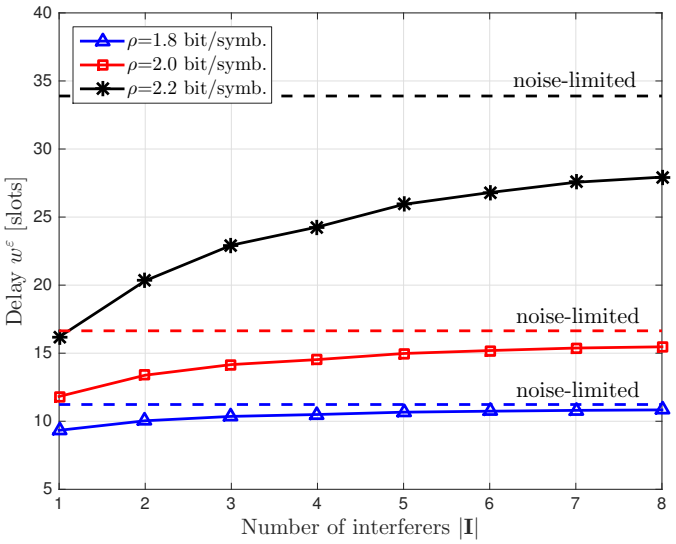


Fig. 3. Delay bound (w^ε) in slots versus the number of interferers ($|\mathbf{I}|$) for different arrival rates $\rho \in [1.8, 2.0, 2.2]$ with fixed $\bar{\gamma} = 8$ dB, average SNR $\gamma_\emptyset = 15$ dB, and $\varepsilon = 10^{-6}$. The delay for the noise-limited case with average SNR $\gamma_\emptyset = 8$ dB is also shown.

Fig. 2a. Again, we observe that the analytical results provide a reasonable bound for the simulated system performance. The figure also shows that the bound accurately predicts the system stability region. Therefore, we conclude that w^ε is a reasonable upper bound for the system's delay performance. In the rest of this section, we focus only on the analytical delay bounds to study different trends of the system performance.

B. Effect of the Number of Interferers

Compared to state-of-the-art networking models that view interference as an additional constant contribution to the noise [6], [7], our explicit consideration of the individual random fading processes of the interferers enables us to address more fundamental questions. One interesting question is the following: Given an average total interference power, what is the impact of the number of interferers on the system performance?

To answer the question above, we show in Fig. 3 the delay as a function of the number of interferers ($|\mathbf{I}|$) in the network for different arrival rates. In this scenario, the average SNR is set to $\gamma_\emptyset = 15$ dB and the delay violation probability to $\varepsilon = 10^{-6}$. We fix the parameter $\overline{\gamma} = 8$ dB, i.e. the sum of the average interference powers stays constant. When there are multiple interferers, the total interference power is distributed among the interferers almost equally without violating the constraint $a_i \neq a_j$ in Eq. (13). However, when the number of interferers grows to infinity, then the combined interference can be modeled as additive noise with constant power as it was demonstrated in [6], [7]. This corresponds to the noise-limited case with average SNR $\gamma_\emptyset = 8$ dB, which we also show for comparison. The figure clearly reveals the importance of the structure of the interference that a certain transmitter/receiver pair is exposed to. In general, if the sum of the average interference powers is kept constant, the more interferers are present, the worse the system performance gets. This happens because in the case of only few interferers, the variance of the SINR is higher, i.e., occasionally the interference is very small, leading to very high transmission rates. In contrast, the higher the number of interferers, the lower is the variance of the SINR, leading a decreasing performance.

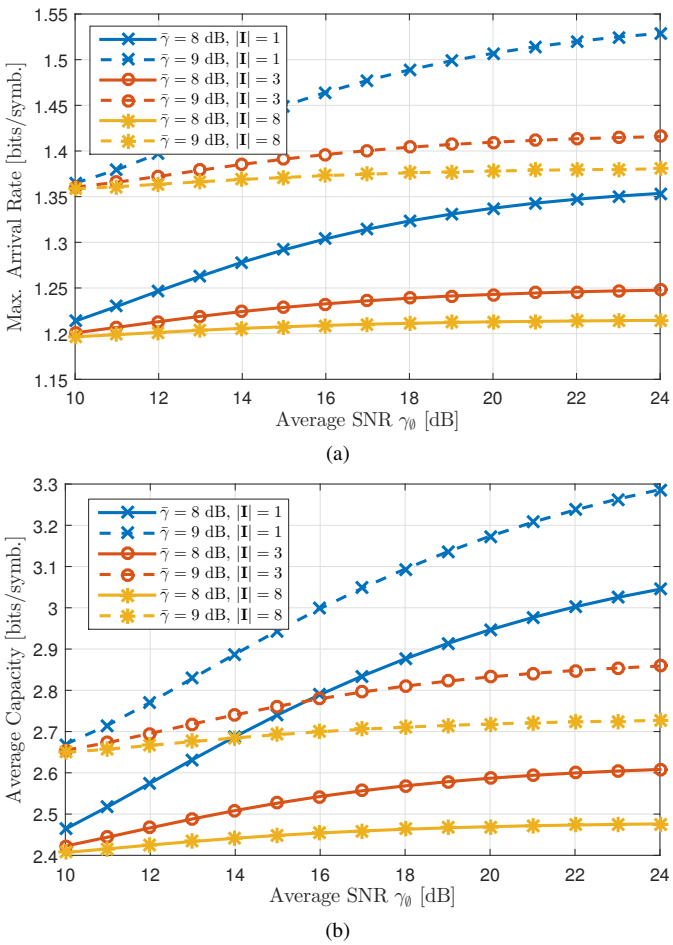


Fig. 4. Effect of average SNR $\gamma_\emptyset$ for different values of $\bar{\gamma} \in \{8, 9\}$ and number of interferers $|\mathbf{I}| \in \{1, 3, 8\}$: (a) Maximum constant arrival rate versus average SNR ($\gamma_\emptyset$) so that delay bound still satisfies $w^\varepsilon = 10$ and $\varepsilon = 10^{-6}$. (b) Average capacity versus average SNR ($\gamma_\emptyset$)

C. Effect of Main Signal Power

Next, we study the effect of the SNR of the signal-of-interest on the system performance. For a system that is operating at a fixed arrival rate and for a given number of interferers, we would like to study the effect of increasing the average signal strength and the interference strength simultaneously, such that the average signal to the average interference plus noise ($\overline{\gamma}$) remains constant.

Fig. 4a confirms the observation that the performance decreases when there are more interferers, despite keeping the summed average interference power constant. Although this was already shown in Fig. 3, we now want to study how this depends on the average SNR ($\gamma_\emptyset$) of the basic signal-of-interest. When the average SNR is small, the disturbance comes mostly from the noise rather than from the interference,

such that the number of interferers has little impact. At high SNR, the noise becomes relatively small, and the performance is limited by the interference. Fig. 4b indicates that the increase in performance for fewer interferers is most likely due to a large increase in the average capacity, which occurs because sometimes the interference-plus-noise can become close to zero, such that there is some probability that the SINR is extremely high. Under delay constraints, it seems that more interferers still decrease the system performance, as seen in Fig. 4a. However, when comparing both figures at high SNR $\gamma_\emptyset$ for different values of $\bar{\gamma}$, it can be seen that higher average capacity does not always mean better performance under delay constraints. Fewer interferers lead to a large increase in average capacity, but only to a small or moderate increase in performance under delay constraints.

V. Conclusions

In this paper, we considered interference channels, where the signal of interest and the signals of an arbitrary number of interferers experience independent Rayleigh fading. We provided a fundamental stochastic characterization of the time-varying channel capacity by its Mellin transform. Using the transform domain and network calculus queuing relations, we contributed the first higher layer performance evaluation of such channels which enabled us to reveal key aspects of interference channels. We showed that even for a fixed summed interference power, the interference channel has relevant degrees of freedom that impact the delay performance significantly, namely strength and number of interfering transmitters. While our evaluations have shown this result for scenarios where the average sum interference power has been kept constant, similar conclusions can be drawn if the average SINR of the scenario is kept constant. Even in this case, the structure of the interference has a significant impact on the performance of the system. As future work, we consider in particular the coupling of wireless systems as key next step that result from the work presented in this paper.

References

[1] Mass Consultants Ltd., "Estimating the utilisation of key license-exempt spectrum bands." Ofcom, 2009. [Online]. Available: http://stakeholders.ofcom.org.uk/binaries/research/technology-research/wfiutilisation.pdf

[2] V. Cadambe and S. Jafar, "Interference alignment and degrees of freedom of the k-user interference channel," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3425–3441, Aug. 2008.

[3] D. Gesbert, S. Hanly, H. Huang, S. Shamai, O. Simeone, and Y. Wei, "Multi-cell MIMO cooperative networks: A new look at interference," *IEEE J. Sel Areas on Commun.*, vol. 28, no. 9, pp. 1380–1408, Dec. 2010.

[4] B. Zhang, W. Cheng, L. Sun, X. Cheng, T. Znati, M. A. Al-Rodhaan, and A. Al-Dhelaan, "Queuing modeling for delay analysis in mission oriented sensor networks under the protocol interference model," in *Proc. ACM MiSeNet*, 2013, pp. 11–20.

[5] N. Bisnik and A. A. Abouzeid, "Queuing network models for delay analysis of multihop wireless Ad Hoc networks," *Ad Hoc Networks*, vol. 7, no. 1, pp. 79–97, 2009.

[6] L. Musavian, S. Aissa, and S. Lambotharan, "Effective capacity for interference and delay constrained cognitive radio relay channels," *IEEE Trans. on Wireless Commun.*, vol. 9, no. 5, pp. 1698–1707, May 2010.

[7] M. Elalem and L. Zhao, "Effective capacity and interference analysis in multiband dynamic spectrum sensing," *Communications and Network*, vol. 5, no. 2, pp. 111–118, May 2013.

[8] L. B. Le, E. Modiano, C. Joo, and N. B. Shroff, "Longest-queue-first scheduling under SINR interference model," in *Proc. ACM MobiHoc*, Sept. 2010, pp. 41–50.

[9] D. Parruca and J. Gross, "On the interference as noise approximation in ofdma/lte networks," in *Proc. IEEE ICC*, Jun. 2014.

[10] H. Al-Zubaidy, J. Liebeherr, and A. Burchard, "A (min, $\times$) Network Calculus for Multi-Hop Fading Channels," in *Proc. IEEE INFOCOM*, 2013, pp. 1833–1841.

[11] R. L. Cruz, "A calculus for network delay. I. network elements in isolation," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.

[12] C.-S. Chang, *Performance Guarantees in Communication Networks*. Springer-Verlag, 2000.

[13] J.-Y. Le Boudec and P. Thiran, *Network Calculus. A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, 2001.

[14] F. Ciucu, A. Burchard, and J. Liebeherr, "Scaling properties of statistical end-to-end bounds in the network calculus," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2300–2312, Jun. 2006.

[15] M. Fidler, "An end-to-end probabilistic network calculus with moment generating functions," in *Proc. of IEEE IWQoS*, Jun. 2006, pp. 261–270.

[16] Y. Jiang and Y. Liu, *Stochastic Network Calculus*. Springer, 2008.

[17] F. Ciucu and J. Schmitt, "Perspectives on network calculus - no free lunch but still good value," in *Proc. ACM SIGCOMM*, Aug. 2012, pp. 311–322.

[18] M. Fidler and A. Rizk, "A guide to the stochastic network calculus," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 92 – 105, Mar. 2015.

[19] Y. Jiang and P. J. Emstad, "Analysis of stochastic service guarantees in communication networks: A server model," in *Proc. IEEE IWQoS*, Jun. 2005, pp. 233–245.

[20] M. Fidler, "A network calculus approach to probabilistic quality of service analysis of fading channels," in *Proc. IEEE GLOBECOM*, Nov. 2006, pp. 1–6.

[21] C. Li, H. Che, and S. Li, "A wireless channel capacity model for quality of service," *IEEE Trans. Wireless Commun.*, vol. 6, no. 1, pp. 356–366, Jan. 2007.

[22] K. Mahmood, A. Rizk, and Y. Jiang, "On the flow-level delay of a spatial multiplexing MIMO wireless channel," in *Proc. of IEEE ICC*, Jun. 2011.

[23] M. Fidler, R. Lübben, and N. Becker, "Capacity-delay-error-boundaries: A composable model of sources and systems," *IEEE Trans. Wireless Commun.*, vol. 14, no. 3, pp. 1280–1294, Mar. 2015.

[24] D. Wu and R. Negi, "Effective capacity: a wireless link model for support of quality of service," *IEEE Trans. Wireless Commun.*, vol. 2, no. 4, pp. 630–643, Jul. 2003.

[25] H. Al-Zubaidy, J. Liebeherr, and A. Burchard, "Network-layer performance analysis of multihop fading channels," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 204–217, Feb. 2016.

[26] B. Davies, *Integral Transforms and Their Applications*. Springer-Verlag, 1978.

[27] S. Kandukuri and S. Boyd, "Optimal power control in interference-limited fading wireless channels with outage-probability specifications," *IEEE Trans. Wireless Commun.*, vol. 1, no. 1, pp. 46–55, Jan. 2002.

[28] A. Polyanin and A. Manzhirov, *Handbook of Mathematics for Engineers and Scientists*. CRC Press, 2006.

[29] N. Petreska, H. Zubaidy, R. Knorr, and J. Gross, "On the recursive nature of end-to-end delay bound for heterogeneous wireless networks," in *Proc. IEEE ICC*, Jun. 2015.

Investigating Packet Loss in Mobile Broadband Networks under Mobility

Džiugas Baltrūnas, Ahmed Elmokashfi, Amund Kvalbein*, Özgü Alay
Simula Research Laboratory, Oslo, Norway
*Nexia, Oslo, Norway

***Abstract*—Mobile broadband (MBB) connections are often exposed to varying network conditions under mobility scenarios, which can result in packet loss and higher end-to-end delays. Such performance degradation in turn can adversely impact the user experience. In this paper, we study packet loss characteristics of MBB networks under mobility using six measurement nodes that are placed on regional and inter-city trains in Norway for a period of seven months. Our findings show that packet loss is significantly higher for mobility scenarios compared to the stationary. In order to understand the cause of packet loss, we investigate Radio Access Technology (RAT) changes, temporary loss of service, and changes in cells and location area codes (LAC). We surprisingly find that almost all periods with RAT changes involve packet loss. We also observe that 70% of the overall loss happens in periods with RAT changes or temporary loss of service. Further, one third of RAT changes involve connection termination. Our findings highlight the importance of radio access network (RAN) planning and configuration, and provide guidelines to alleviate packet loss in MBB networks.**

I. Introduction

Mobile broadband is becoming the primary Internet access method for a large number of people and services. All types of Internet applications (office, games, video, web, cloud) are now accessed over MBB. According to the Cisco VNI Global Mobile Data Traffic Forecast, global mobile traffic grew by 69% in 2014 and is expected to grow almost tenfold by 2019 [7]. This tremendous growth in MBB demand has put understanding and improving its performance high on the agenda of both decision makers and industry. Several governments have launched activities to measure MBB performance [1, 8]. Measuring and understanding MBB performance is, however, a challenging task. The plethora of scenarios an MBB connection typically experiences requires context-specific studies. Further, there is a lack of measurement methodologies and metrics that are tailored specifically for assessing MBB [4].

The ability to deliver data packets as reliably as possible is arguably one of the most important quality metrics in MBB networks. Excessive and even sporadic packet loss worsens the user experience significantly. It degrades the performance of reliable transport protocols, increases retransmissions, and ultimately degrades application performance. Assessing and mitigating packet loss is an important step for improving MBB performance. There are, however, many potential causes of loss, which makes characterizing and understanding loss a non-trivial task [5]. This task becomes particularly daunting under mobility. Moving connections experience varying signal quality, cellular handovers, potential changes in radio technology, just to name a few.

ISBN 978-3-901882-83-8

In this work, we perform a longitudinal practical investigation of packet loss under mobility in operational MBB networks using end-to-end measurements. To this end, we use over half a year's worth of measurements from six measurement nodes that are placed on board regional and inter-city trains in Norway. We use this data to compare and characterize loss under mobility to stationary scenarios. We further leverage connection state information to identify the underlying causes of loss. Our measurements and analysis give insights into the characteristics and causes of packet loss under mobility. In summary, this work makes the following contributions:

1) We present the most comprehensive study of loss in MBB networks under mobility. Using over half a year's worth of measurements and data points from diverse geographic locations, we are able to pinpoint causes of loss under mobility and derive a classification methodology.
2) We demonstrate that performing end-to-end active measurements in conjunction with collecting connections' metadata can help dissecting the most complex MBB scenarios.
3) Our results single out technology handovers and coverage holes as the main causes of loss under mobility. We use these insights and other findings to identify potential areas for improvement.

The rest of this paper is organized as follows. Section II presents the measurement scenario and data. Section III discusses basic statistics of loss under mobility and proposes a classification methodology to link loss to its likely causes. Sections IV and V analyzes loss in periods with and without connection technology changes respectively. We highlight the related work in Sec. VI and conclude in Sec. VII.

II. Scenario and data

A. Measurement Setup

The measurement setup used in this study consists of six measurement nodes placed on six regional and inter-city trains operating in Norway. This paper is based on data collected by these nodes from July 2014 until February 2015. Figure 1 shows the routes covered by these trains, which includes a reasonable mix of urban and rural areas.

This measurement setup is the mobile subset of the NorNet Edge (NNE) [13], which is a country-wide measurement infrastructure that consists of several hundreds nodes for

Fig. 1: Map of the train routes

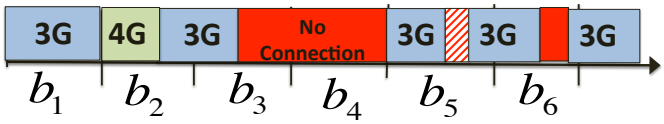


Fig. 2: Typical sequence of connectivity and coverage conditions that MBB connections experience as they move.

measuring the performance and reliability of MBB networks. NNE nodes are single board computers that run a standard Linux distribution and connect to up to four Universal Mobile Telecommunications System (UMTS) operators and one Code-Division Multiple Access (CDMA) 1xEV-DO operator. Like other NNE nodes, train nodes connect to up to four UMTS operators via Huawei E392-u12 modems that support up to Long Term Evolution (LTE) Category 3, and one CDMA 1xEV-DO operator. In this study, however, we limit ourselves to studying two UMTS operators, Telenor and Netcom, because these operators are the only operators that provide LTE service and run their own radio access and core networks. Software running on NNE nodes ensures that MBB connections are always alive, and collects connection state information. In particular, we monitor and record the RAT, which can be *No service*, 2G, 3G or LTE; different signal quality indicators (e.g., RSSI, Ec/Io, and RSRQ); network attachment information (e.g., serving cell identifier, location area code, and tracking area code); and Radio Resource Control (RRC) state. To measure packet loss, we send a 20-byte UDP packet every second over each connection to an echo server that is part of NNE backend and then record a reply packet from the server. A packet is considered lost if we do not receive a reply within one minute. Further, we aggregate the data into five minute bins and calculate loss percentage for each bin. Both measurement data and metadata are periodically transferred to a server and imported into a database. We also use the GPS location data from the train's fleet management system to identify the location of NNE measurement nodes and trains speed during the measurements. The GPS locations are updated every 10 to 15 seconds in the fleet management system.

B. Measurement scenario

Figure 2 shows a typical sequence of connectivity and coverage conditions that MBB connections experience as they move. RAT can be constant during a whole bin or several consecutive bins as in b_1. A bin may involve several horizontal handovers, that is changes of the serving cell. Some bins involve inter-RAT handovers (e.g., handover from 4G to 3G in b_2). All types of handovers are well defined procedures that should normally last a couple of seconds and degrade the user experience negligibly. Connections may suffer from lack of coverage, which leads to a complete loss of connectivity for several minutes (e.g., no connection period extends from the mid of b_3 to the end of b_4). In these periods, a connection looses its Packet Data Protocol (PDP) context (3G) or Evolved Packet System (EPS) bearer (LTE), which is a tunnel that connects the user equipment (UE) to the core network (CN). Consequently the connection looses its IP address. When a connection breaks, software on the nodes immediately checks if there is coverage and tries to reconnect. Otherwise it waits until coverage becomes available. Connections may also experience temporary loss of connectivity that is immediately rectifiable e.g., the short disconnection during b_6. These episodes can be caused by temporary lack of coverage (i.e., coverage holes) or due to the interplay between mobility patterns and handover procedure decision and duration. For example, the modem may start a handover to a new cell, a procedure that involves current and neighbor cells, but it looses connectivity to the current cell before completing the handover. Another cause can be failures during inter-RAT handovers, which are known to happen [14]. Finally, connections also experience periods with brief lack of service that are immediately followed by a service restoration without inter-RAT handover or context reset like the shaded area during b_5. Note that during periods with lack of service, connections typically remain attached to the network and appear to have a PDP context (EPS bearer).

In this paper, we are interested in measuring users experience as nodes move and have connectivity. Accordingly, we divide the measurement bins into two groups. 1) bins where users experience lack of coverage, and 2) bins where users have coverage but may experience brief lack of connectivity that is restored by an immediate reconnection attempt. All bins in the first category (58% and 54% of bins in Telenor and Netcom, respectively) are discarded in the remainder of this study.

C. Data curation

UE and connection managers always try to cope with the varying coverage and connectivity conditions by quickly detecting lack of connectivity, attempt to reconnect, or reset the wireless device altogether. The interplay between varying signal conditions and UE hardware is non-trivial and it may sometimes render the connection unusable. We believe that

some failure situations are caused by specific measurement and system artifacts; a different system or hardware may cope better or worse. Next we describe these artifacts in more details.

Sometimes modems become unresponsive and are eventually ejected by the operating system, resulting in a disconnect and probably packet loss before the ejection. In some other cases, the PDP context (EPS bearer) might seem to be operational, but IP packets cannot be sent or received until the connection is re-established. We refer to these connections as stale connections. We verify that connections become stale when the network attempts to reset long-lived PDP contexts (EPS bearers), and it fails half-way through the process without actually reseting the context (bearer). As a result, the operator's firewall drops all incoming packets from these connections, causing 100% packet loss during these periods.

Other artifacts include server-side failures and measurements with misreported metadata. For example, the modem reports that it is on LTE while at the same time reporting 3G-specific metadata such as Ec/Io or RSCP. This typically happens when the modem delays sending metadata because it is busy with processing control traffic.

To be able to cope with the aforementioned anomalies, we impose a number of filters to the dataset. This leaves us with only measurement data that is supported by clean metadata. Next, we describe our filters:

1) We remove all 5-minute bins with 100% packet loss to avoid stale connections and cases where the modem is stuck and yet appears operational to the OS. By doing that, we risk excluding some legitimate loss events that are caused by equipment failures and maintenance activity [5]. These events are, however, outside the scope of this study since we are interested in what users experience on a daily basis and not rare or scheduled events.
2) We look only at bins where the train was moving, and we require at least one available GPS reading in a 5-minute bin in order to determine this. We impose the average speed of the available readings to be > 0. To check for the cases when the train was predominately still during a 5-minute period, we imposed larger average speed thresholds and observed similar results.
3) We remove all 5-minute bins that coincide with known server-side maintenance.
4) We keep only 5-minute bins where we have metadata reports for at least 4 of the 5 minutes These reports are acquired by polling the modem at the beginning of each minute.
5) We keep only bins with known RATs and valid combination of RAT and RAT-specific metadata.

After curating the initial data set, we have 63837 five minute bins, 38417 from Telenor and 25420 from Netcom.

III. Loss under mobility

In this section, we give a general overview of loss characteristics in mobile networks. We investigate the effect of mobility, and establish the very different loss rates in a mobile vs stationary scenario. We also look at loss in different RATs, before we proceed to classify loss under mobility and relate it to handover events.

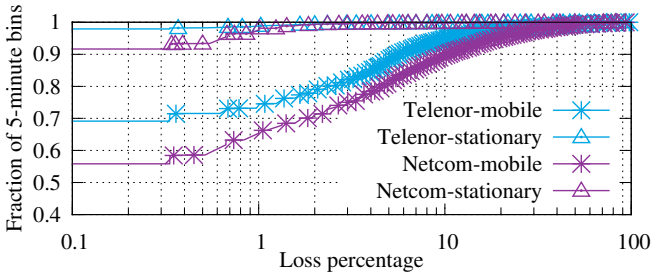


Fig. 3: Overall loss rate and the effect of mobility. Much higher loss rates observed when mobile in both networks.

A. The effect of mobility

Figure 3 shows the overall loss rate for the two measured networks, when the nodes are stationary and moving. It is clear from the figure that loss is much higher when mobile than when stationary. When the nodes are stationary, only 2 % (Telenor) to 12 % (Netcom) of 5-minute bins involve packet loss. In the mobile case we observe loss in 30% (Telenor) to 50% (Netcom) of bins, and 5% (Telenor) to 10 % (Netcom) of bins have a loss rate above 10%. This large difference in loss rate between mobile and stationary nodes largely motivates this study. In our previous work [5], we analyzed loss in a stationary scenario. We concluded that loss rates are low in general and that causes are related to misconfiguration of the radio access controller or lie beyond the RAN. In the remainder of this paper, we will dissect and seek to explain what causes loss to be so much higher under mobility. In most of our analysis, we do not differentiate between movement at different speeds, since we observe that this has a limited effect on packet loss (see Sec. V). Only less than 4% of our 5-minute bins have the average speed of 100 km/h or more, whereas it has been shown that the effect of speed alone on packet loss is negligible for train speeds below 150 km/h [14].

B. Loss in different RATs

The measurement nodes used in this study will always try to connect to the highest available RAT. That is, they will prefer LTE over 3G over 2G. To investigate loss in different RATs, 5-minute bins are divided into 3G, LTE and mixed. Mixed bins are bins where the node was connected to more than a single RAT. We do not include bins spent fully on 2G in our analysis, since there are relatively few such bins, and both loss rates and connection stability are much worse in this RAT. 2G bins are experienced mostly in challenged areas (with limited coverage), and that our measurements are therefore not representative for normal 2G behavior. In particular, our dataset contains only 190 and 148 2G bins for Telenor and Netcom, respectively. In these bins, the average loss rate is between 17% and 18.4%, while the median loss rate varies from 11.6% to 13%.

Figure 5 shows loss for different RATs when the measurement nodes are moving. We first observe that loss rate is higher in 3G than in LTE. Less than 4 % of LTE-only bins experience

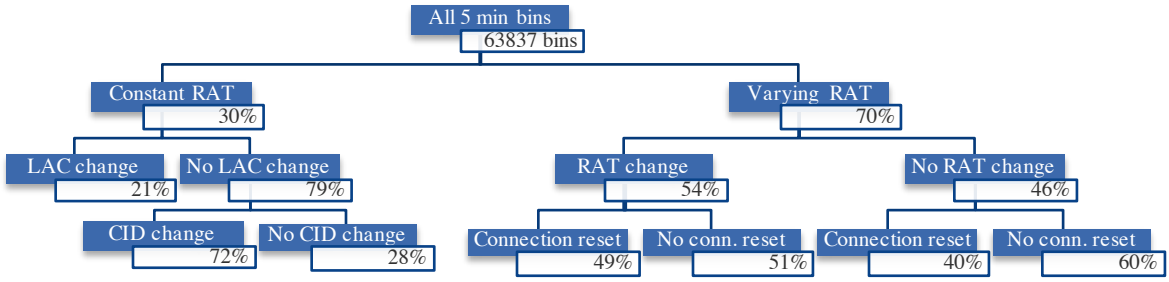


Fig. 4: Classification of loss. The numbers given are percentages of lost packets relative to the parent category. In total, there are 63837 5-minute bins in the dataset. About 229992 packets were lost during these bins.

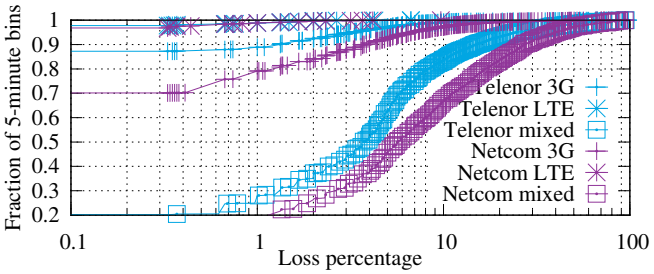


Fig. 5: Loss rate for 3G, LTE and when a RAT change is involved (mixed). Most loss happens in mixed category, LTE performs the best.

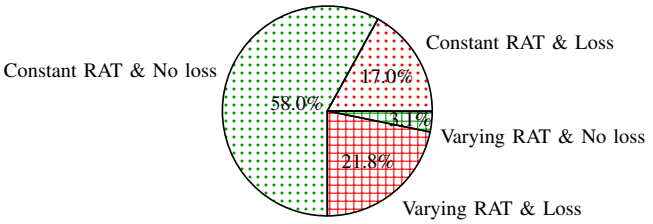


Fig. 6: The percentages of lossy and non-lossy 5-minute bins for constant and varying RATs.

loss, while 14 % (Telenor) to 40 % (Netcom) of 3G-only bins experience loss.

The most striking observation from Fig. 5 is, however, the much higher loss rate in mixed bins. 86% (Telenor) to 94% (Netcom) of bins with RAT changes also have packet loss. In 16 % (Telenor) to 33% (Netcom) of bins, the packet loss is over 10%. This indicates that inter-RAT handovers is a major source of loss in MBB networks. We perform an in-depth analysis of this loss in Sec. IV.

C. Classification of loss under mobility

In order to structure our investigation of loss, we start by classifying all 5-minute bins according to the state of the connection in that bin. We perform this classification in a hierarchical fashion, as shown in Fig. 4. This classification captures the connectivity states shown in Fig. 2 and isolates independent conditions, thus reducing the complexity of identifying potential causes of loss.

The root of the tree contains all bins where the measurement nodes have radio coverage as discussed in Sec. II. Inspired by the observation in Fig. 5 that loss is much higher when there is a RAT change, we first split the bins into constant and varying RAT. Constant RAT bins are characterized by a single RAT throughout their duration. Varying RAT bins, however, involve more than one RAT or a short lack of service. 30% of all loss occurred during bins with a constant RAT, while in 70% of loss occurred during bins where the RAT changed at least once. Loss rate is on average seven times higher in bins with varying RAT.

Figure 6 shows the distribution of lossy and non-lossy bins for constant and varying RAT cases. The percentage values shown are relative to all bins (they add up to 100%). We observe that a clear majority (more than $3/4$) of bins with constant RAT experience no loss. On the other hand, almost all bins with varying RAT involve packet loss.

Bins with varying RAT are further divided into bins where the connection is attached to at least two RATs, and those where we only observe one RAT. Note that a bin might still be classified as varying even if we only observe one RAT: this means that the modem reported no available RAT at least once during the bin. This behavior is a normal part of a RAT transition, but it also sometimes appears without a resulting RAT change.

Recall from Sec. II that we only include 5-minute bins without connection resets, or where a single reconnection attempt immediately restores connectivity. The varying bins with and without RAT changes are further subdivided according to whether there is a connection reset in the bin. We separate bins with connection resets because we believe that they are characteristically different from the rest of the varying bins. These resets are likely caused by small coverage holes and failures of the handover procedure. We observe from Fig. 4 that all varying RAT categories are responsible for a significant share of the overall loss. Loss in bins with varying RAT is further explored in Sec. IV.

The constant RAT bins are further classified according to whether there is a LAC change in the bin, and if not so, whether there is a cell ID (CID) change. The intuition behind this classification is a hypothesis that horizontal handovers (change of LAC or CID) is an important source of loss in constant RAT periods. The numbers in Fig. 4 confirms that this is the case: 78% of loss in constant RAT periods happen in bins with either LAC or CID changes. Section V provides a more detailed investigation of causes of loss in constant RAT periods.

IV. Varying RAT

In this section, we analyze loss in bins with varying RAT, which constitutes 25% of all bins and are responsible for 70% of the overall loss. As mentioned in Sec. III, bins with varying RAT come in two forms. First, bins with one or more inter-RAT handovers; that is, we observe more than a single RAT in the bin. Second, bins with no handovers but a glitch in the service. The second class of bins are characterized by one predominant RAT (2G, 3G or LTE), but with the presence of one or more *No service* episodes throughout the bin. Figure 7

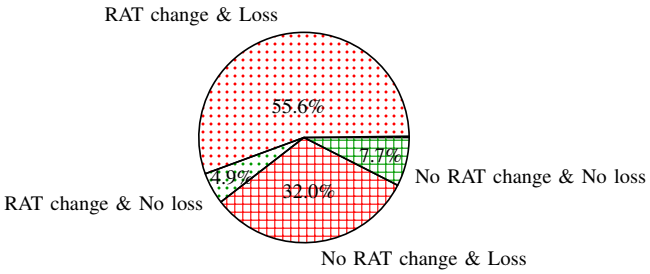


Fig. 7: The percentages of lossy and non-lossy 5-minute bins with varying RATs split by whether more than one RAT is observed.

divides bins with varying RAT according to the presence of RAT changes and loss. A large majority of bins with inter-RAT handovers, about 92%, involve packet loss. This fraction is slightly smaller, about 80%, for bins with no RAT change. In the following subsections, we will look at both scenarios and investigate possible causes of loss.

A. Loss during periods with RAT changes

The modems in the setup are configured to automatically select the highest available RAT. In large cities, LTE is almost always available, while outside the metro areas, 3G is the predominant RAT. Inter-city trains cross rural areas where conditions can vary from strong 3G to weak 2G signal to no coverage. RAT changes, or inter-RAT handovers are based on the UE neighbor cell measurement reports, which are regularly sent to the network. Based on these reports, the network can initiate a handover from one RAT to the other. This typically happens when measurements show that the signal and the interference levels from the current cell in RAT A are worse compared to levels of RAT B. It can also happen when the UE moves into the range of a new cell or cell sector that supports RATs different from the current RAT. The handover process involves multiple steps both in the UE, the RAN and in the CN. These steps vary depending on the type of handover, e.g., from 3G to 2G, from LTE to 3G, etc. Further, inter-RAT handovers are not always seamless. For example, sometimes we observe RAT changes that lead to a connection reset. Moreover, it is quite common to lose several packets right before the connection breaks.

Figure 8 shows the split of bins containing an inter-RAT handover, according to the presence of loss and connection resets. The main observation is that bins with inter-RAT handovers involve packet loss independent of whether there is a connection reset or not. Almost all (99%) bins with RAT

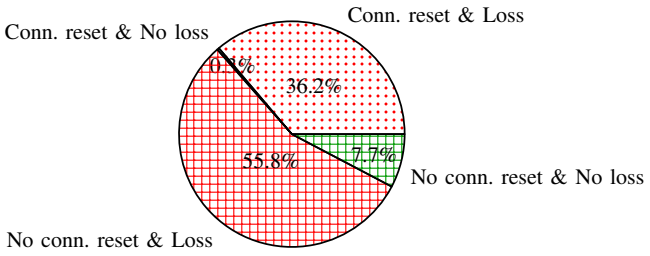


Fig. 8: The percentages of lossy and non-lossy 5-minute bins for varying RAT with a RAT change split by the presence of connection resets.

change and connection resets include packet loss compared to 88% of the bins with RAT changes but no connection resets. The majority of RAT changes, about two thirds, complete without a connection reset. Overall, *inter-RAT handovers are patently lossy and involve a short loss of connectivity in over one third of the bins*.

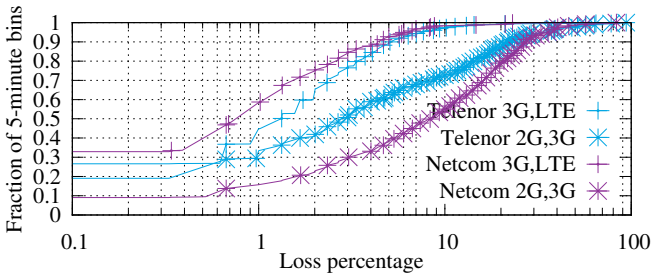


Fig. 9: Involved RATs for the 5-minute bins with a RAT change and no connection resets. Clear differences between networks. More loss during 2G/3G handovers.

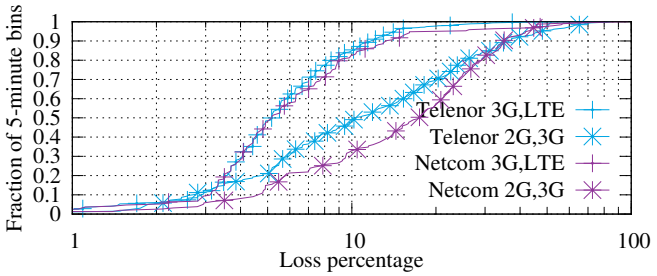


Fig. 10: Involved RATs for the 5-minute bins with a RAT change and one or more connection reset. Much more loss compared scenario without disconnects, minimal differences between networks, least lossy are bins with 3G/LTE handovers.

To further analyze loss related to RAT changes, we identify all distinct RATs present in each 5-minute bin with an inter-RAT handover [1]. Figures 9 and 10 shows the distribution of loss in bins with RAT changes split by the involved RATs. Figure 9 shows bins *without* connection resets, while Fig. 10 shows bins *with* connection resets.

The plots highlight three interesting facts:

1) We only observe *minor difference between Netcom and Telenor in both plots*. This suggest that the same underlying causes lead to this loss in both networks.

2) *The majority of bins with RAT changes include packet loss regardless of the involved RATs.* Loss is, however, much higher

[1]There can be more than one handover in a 5-minute bin, which means that there is no one to one mapping between the number of distinct RATs and the number of handovers.

when 2G is among the distinct RATs. Bins where 2G is involved occur in poorly covered areas that are characterized by coverage gaps and the dominance of 2G.

3) *Packet loss in bins with connection resets is markedly higher*; loss is over 3% in 90% of the bins. We believe that this loss is a consequence of unsuccessful handovers, which initially result in packet loss followed by the connection reset.

To quantify the impact of loss in bins with RAT changes, we count the number of consecutively lost packets in each loss episode. This captures the burstiness of packet loss and we term such consecutively lost packets a loss run. We observe that for the bins without connection resets, most loss runs are of size one. However, loss run size is characteristically different for the bins that involve one or more connection reset. Figure 11 shows the probability density function of the loss run size distribution for different RAT combinations for bins with connection resets.

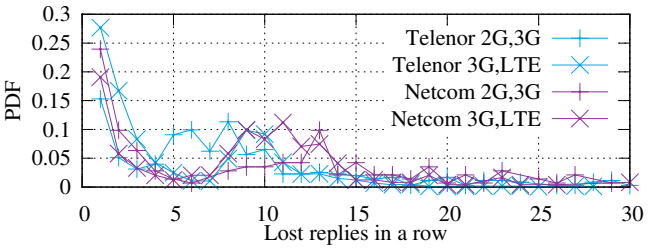


Fig. 11: Loss runs for involved RATs when there is a RAT change and one or more connection reset. Several modes around 10 packets in both networks. Different modes for different sets of involved RATs.

While a sizable fraction of loss is still random regardless of the network or involved RATs, there are some modes at loss runs of size 5 to 13. Since these modes are present in both networks and in the two sets of RATs, we believe that they are caused by the response of the specific UE to weak or failing coverage.

The loss runs in bins with connection resets consist of two components. First, loss that happens right before the connection reset; we often experience degraded performance before a connection reset. Second, loss that happens between the actual loss of PDP context (EPS bearer) and until PPPd discovers the loss of IP address. We typically detect the loss of connectivity immediately. This detection, however, may take much longer if the modem stops communicating with the PPPd by not responding to PPP echo requests regularly sent by the daemon. These cases can occur when the modem is busy with trying to exchange signaling messages with the network. PPPd on the measurement nodes responds to the lack of echo replies by tearing down a stuck connection after six seconds. This partially explains the mode around six in Fig. 11.

B. Loss during periods with RAT glitches

In case RAT becomes unavailable, the modems report a special RAT called *No service*. In some cases this RAT is also reported during the inter-RAT handover procedure. *No service* periods mostly happen during the temporary loss of coverage and/or unsuccessful inter-RAT or horizontal handovers. In this subsection, we investigate loss in bins that include strictly one RAT and at least one *No service* period.

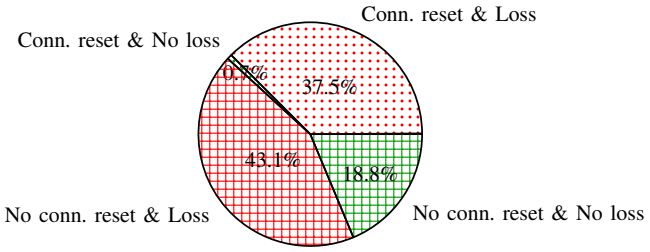


Fig. 12: The percentages of lossy and non-lossy 5-minute bins for varying RATs with no RAT change split by the presence of connection resets.

As for the periods with inter-RAT handovers, some of the bins with no RAT changes have one or more connection resets. Figure 12 shows the split of varying RAT bins with no RAT change according to the presence of loss and connection resets. The overall percentage of lossy bins is slightly smaller (81.6%) compared to the scenario with RAT changes, but it is still high. There are also more (18.8%) non-lossy bins without connection resets compared to the scenario with RAT changes.

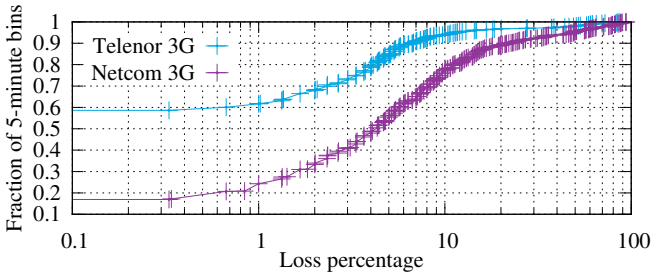


Fig. 13: Individual varying RATs for the 5-minute bins when RAT does not change and there are no connection resets. Less loss in 3G bins with varying RAT in Telenor compared to Netcom.

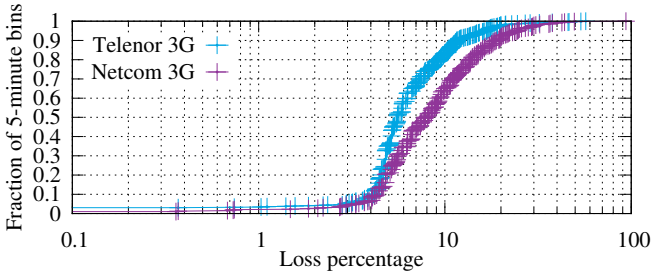


Fig. 14: Individual varying RATs for the 5-minute bins when RAT does not change and there is one or more connection reset. Differences between networks are minimal, but there is slightly less loss in 3G bins with varying RAT in Telenor compared to Netcom.

Next, we look at loss rate distributions for bins with *No service*. Figure 13 and 14 show the loss rate for the bins without or with one or more connection resets, respectively.

Here we focus on 3G bins only, since the number of 2G and LTE bins is very low in both networks. This can be explained by the fact that 3G is the dominant RAT country-wide and therefore most handovers happen on 3G. Telenor exhibits much less loss compared to Netcom in bins without connection resets, hinting that coverage problems that lead to *No service* are less prevalent in Telenor. This matches well

our out-of-band understanding of the coverage of these two operators; Telenor has a denser deployment of cell towers than Netcom. Loss in bins with connection resets is, however, much higher and very similar across the two networks. We believe this similarity is a product of the non-trivial response of UE to the loss of coverage as explained in Sec. IV-A.

To quantify the impact of loss in bins with RAT changes, we now look at the distribution of loss runs sizes. Figure 15

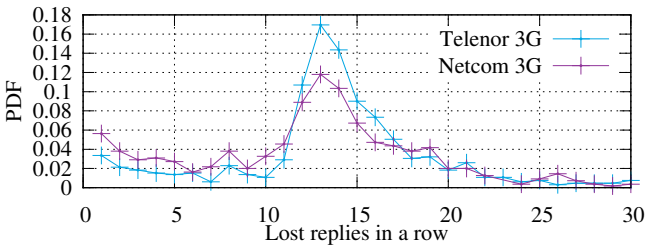


Fig. 15: Loss runs for individual varying RATs when there is no RAT change, but one or more connection reset in a 5-minute bin intersecting with the loss run. All modes in a shorter range from 11 to 16, with a peak at 13 packets for both networks.

shows the PDF of the loss run size distribution for the two networks. These distributions clearly differ from the loss run size distribution for bins with RAT changes and connection resets in two respects. First, there is no random loss. Second, loss run sizes are confined to a narrow range between 10 and 15. These observations indicate that these loss runs must be triggered by temporary lack of coverage followed by the connection resets.

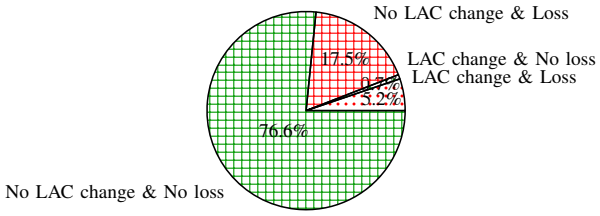
Summary of findings. This section has shown that the loss rates are high in periods with varying RAT, independent of whether there is an actual inter-RAT handover or not. About 40% of bins with varying RAT also contain a connection reset. If connection resets are involved, we normally also see packet loss, and the loss episodes are more severe.

V. Constant RAT

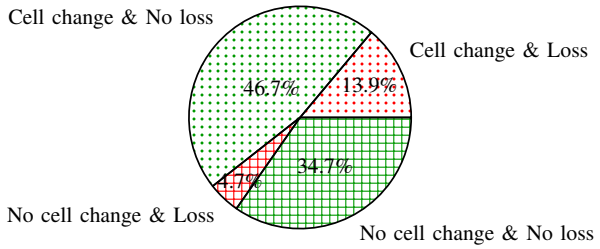
This section investigates bins that are characterized by constant RATs (i.e., no inter-RAT handovers), which are located on the left most subtree in Fig. 4. During these periods a connection may experience LAC and cell changes as well as channel quality degradation.

As shown in Sec. III, about 30% of packet loss during mobility takes place in bins with *Constant RAT*. Most of this loss, 72%, coincides with changes of serving cells (CID change).

Figure 16a divides *Constant RAT* bins based on whether there is a LAC change or not and shows the percentage of bins that fall into four different categories that describe LAC change and loss. The fraction of bins with LAC changes is small (6.3%), which is expected since one LAC mostly covers large geographical areas and LAC changes happens when crossing the boundaries between areas. Connections used in this study experience loss in 88% of the bins with a LAC change. For a smooth handover, the UE needs to be able to communicate with both the current and the candidate cells upon starting the handover procedure. Once the handover is



(a) LAC change



(b) Cell change

Fig. 16: The percentages of lossy and non-lossy 5-minute bins for constant RATs split by the LAC and cell changes.

completed, in-flight packets will be re-routed to the new cell. Inter-LAC handovers are slightly more challenging, since they involve additional coordination between several RNCs [2], i.e., the handover procedure takes longer to complete compared to cell changes within the same LAC.

Figure 16b divides *Constant RAT* bins without LAC changes into four categories that capture both cell changes and loss. The connections experience a cell change in 60% of all bins without LAC changes. These handovers are usually smooth, with 77% completing without a single packet lost. Loss in bins with cell changes is, however, three times higher compared to those without. Figure 17 shows loss rate distribution for bins with LAC changes, cell changes, and no changes when the connections are on 3G or LTE.

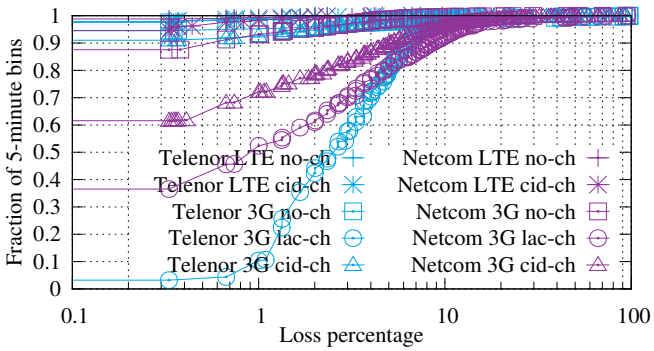


Fig. 17: Loss rate in a 5-minute bin for 3G and LTE RATs split by the presence of one or more LAC change or cell change, or none of them. In both networks the highest loss occurs when there is a LAC change involved. In Telenor, bins with LAC changes are more lossy compared to Netcom, whereas in Netcom there is much loss during cell changes. When there is no LAC or cell change or the LTE cell changes, the loss is minimal in both networks.

Loss rates are evidently higher in bins with LAC changes with clear differences between operators. Almost all LAC

[2]In theory, an RNC may serve more than one LAC. Private communications with the measured operators confirmed that is not the case in the networks we measure.

changes in Telenor involve packet loss, while for Netcom, LAC changes seem to be smooth in 40% of the cases. Hence, this loss appears to be dependent on the network configuration. We also observe that Netcom 3G connections experience significantly higher loss when switching cells compared to Telenor 3G connections. Loss is minimal during LTE cell changes with no clear differences between operators. We believe that loss during handovers can happen due to one of the following three reasons:

1) Short coverage gaps between adjacent cells.
2) Misconfigured neighbor cell list, which makes affected cells not aware of their neighbors and thus unable to complete handovers successfully.
3) A complex interplay between the timing of the handover decision and trains speed. When deciding to handover, the UE performs an attachment procedure during which it becomes attached to two cells; the current cell and the candidate cell. The handover will break, if the UE looses sight of the old towers during the movement while the procedure is ongoing.

[labelindent=0pt,itemindent=0pt]

Scenario	S<10	10<S<50	50<S<100	100<S
CID-3G	0.48	0.45	0.60	0.72
CID-LTE	0.10	0.21	0.67	NA
LAC-3G	0.85	0.75	0.90	0.79

TABLE I: Fraction of lossy bins for Netcom cell and LAC changes for different speed (S) categories. The speeds are in km/h.

Table I shows the fraction of 5-minute bins in Netcom that involve loss for different train speed categories and different horizontal handover scenarios. We choose Netcom because it demonstrates significantly more loss during CID changes. The likelihood of experiencing loss during CID changes evidently increases as the speed increases over 50 km/h. LAC changes, however, involve loss independent of the speed, suggesting that the root cause of loss is perhaps related to inter-LAC handover procedure configuration. Note that we have not measured LTE cell changes when the speed is higher than 100 km/h. Trains reach high speeds outside the metro-area and Netcom seem not to have LTE coverage in these areas.

Summary of findings. Loss is significantly lower in bins where the RAT type is stable. With a stable RAT, cell handovers, and in particular those involving also a LAC handover, is a main cause of loss. There are clear differences in how handovers affect loss between operators.

VI. Related work

There has been a growing interest in performance and reliability measurements of MBB networks. Regulators need measurements to monitor how operators fulfill their obligations, and as a baseline for designing regulatory policies. On the other hand, operators are interested in operational instability and anomalies to identify problems in their networks. There are mainly three approaches for measuring the performance and reliability of MBB networks: (i) crowd-sourced results from a large number of MBB users [2, 15, 19, 23] , (ii) measurements based on network-side data [10, 11, 21, 22] and (iii) measurements collected using dedicated infrastructure [4, 12, 20]. In this paper, we collect data from a dedicated infrastructure in order to have full control over the measurement nodes, allowing us to systematically measure the reliability over a long period of time. The long-term end-to-end measurements lead to a better quality dataset without requiring access to network-side logs, which are typically only available to operators.

Several studies focused on the causes of packet loss in MBB networks. Different groups blamed RRC state transitions [5, 6, 16–18] and showed that the operators do not always configure their RRC state machines according to the standard guidelines leading to significant loss during state demotions. Gember et al. compared packet loss on idle and near active devices and found loss rates on idle devices to be 26% higher and likely to be caused by differences between cell sectors [9]. Xu et al. discussed the effect of bursty packet arrivals and drop-tail policies employed by the operators [25]. RNC-level performance analysis of UMTS networks identified correlations between RTTs and loss and their dependency on diurnal patterns and overloaded NodeBs [6]. One study showed that most transport-layer packet loss is related to physical layer retransmissions and can be reduced by buffering [11]. Another study presented a framework for measuring the user-experienced reliability in MBB networks, and showed how both radio conditions and network configuration play important roles in determining reliability [4]. Both of these studies consider only stationary scenarios, while in this paper we focus on mobility scenarios where signal quality is varying as well as handovers are present.

Packet loss has also been investigated for mobility scenarios. Li et al. [14] studied TCP performance in HSPA+ networks on high-speed rails and showed that the number of handovers is proportional to the increased loss rates for high speeds. Similar observations were made in a study by Balachandran et al. [3], showing that most HTTP sessions with inter-RAT handovers are abandoned. Tso et al. [24] measured HSPA performance on the move to be greatly different from static HSPA performance. In particular, they observed that the final results of handovers are often unpredictable and that UDP packet loss at least doubles during handover periods. Although these studies considered different aspects of packet loss for stationary and mobility scenarios, to the best of our knowledge, there has been no comprehensive study that characterizes packet loss in 3G and LTE networks and compares the mobility and stationary scenarios. Along with the end-to-end measurements used in this work, we further leverage connections' metadata and state information to identify the underlying causes of loss.

VII. Discussion and conclusions

This paper has analyzed the causes of loss in MBB networks under mobility. The observed loss rates are much higher than in the stationary case [5]. In particular, disturbances or handover between different RATs is a main cause of loss, accounting for about 70% of the total. Such RAT changes also often involve a reset of the data connection between the UE

and the network, which mostly involves heavy packet loss. Cell changes are also an important source of loss, and cell changes that also involve a LAC change are the worst.

The observed dominance of loss during RAT changes highlights such handovers as an area that warrant particular attention from mobile operators. The inter-RAT handover procedure is complex, and involves interaction between the UE, the RAN and the CN. The most efficient way to reduce packet loss is to improve the procedures for how such handovers are performed. The number of such handovers should be limited, and packets in transit should be buffered or retransmitted to avoid loss.

There are significant differences between the two networks measured in this study with respect to loss during cell changes. While Telenor experiences significantly more loss during LAC changes, Netcom sees more loss during normal cell changes. These differences indicate that operators still have a significant potential for reducing loss through better configuration settings in their network.

To verify some of our findings, we conducted a drive test in Oslo area by placing the measurement node in a car. In total, we have collected over 5 hours of measurements for the two networks. These measurements confirmed that in Telenor, almost all (92%) packets were lost during the periods with varying RAT. In Netcom, around one half of loss happened in bins with varying RAT too, while the second half was during periods with cell or LAC changes. As expected, we have not observed any case with varying RAT and a temporary loss of service, as it is very unlikely to have coverage holes in the city. In other words, results from the drive test confirm and highlight that inter-RAT handovers are prone to high packet loss even in well covered areas.

End-to-end measurements used in this study are useful for quantifying and characterising the problem. However, to localise the root causes of packet loss, this might not always be sufficient. We therefore acknowledge that network side data or measurements from the RAN could give more insights into the potential causes and assist in improving the network.

VIII. Acknowledgements

This work was partially supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 644399 (MONROE) and by by the Norwegian Research Council under grants 209954 (Resilient Networks 2) and 208798 (NorNet). The views expressed are solely those of the author(s). We would like to thank NSB for hosting measurement nodes aboard their operational passenger trains.

References

[1] *Annual report of the Communications Regulatory Authority (RTT) of the Republic of Lithuania*, 2012.

[2] Mobiperf. http://www.mobiperf.com, 2014.

[3] A. Balachandran, V. Aggarwal, E. Halepovic, J. Pang, S. Seshan, S. Venkataraman, and H. Yan. Modeling Web Quality-of-Experience on Cellular Networks. In *Proc. of MobiCom*, 2014.

[4] D. Baltrūnas, A. Elmokashfi, and A. Kvalbein. Measuring the Reliability of Mobile Broadband Networks. In *Proc. of IMC*, 2014.

[5] D. Baltrūnas, A. Elmokashfi, and A. Kvalbein. Dissecting Packet Loss in Mobile Broadband Networks from the Edge. In *Proc. of INFOCOM*, 2015.

[6] Y. Chen, N. Duffield, P. Haffner, W. ling Hsu, G. Jacobson, Y. Jin, S. Sen, S. Venkataraman, and Z. li Zhang. Understanding the Complexity of 3G UMTS Network Performance. In *Proc. of IFIP Networking*, 2013.

[7] *Cisco visual networking index: Global mobile data traffic forecast update, 2014 - 2019*. Cisco Systems, Inc., February 2015.

[8] FCC. 2013 Measuring Broadband America February Report. Technical report, FCC's Office of Engineering and Technology and Consumer and Governmental Affairs Bureau, 2013.

[9] A. Gember, A. Akella, J. Pang, A. Varshavsky, and R. Caceres. Obtaining In-Context Measurements of Cellular Network Performance. In *Proc. of IMC*, 2012.

[10] E. Halepovic, J. Pang, and O. Spatscheck. Can you GET me now?: Estimating the time-to-first-byte of HTTP transactions with passive measurements. In *Proc. of IMC*, 2012.

[11] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *Proc. of SIGCOMM*, 2013.

[12] Z. Koradia, G. Mannava, A. Raman, G. Aggarwal, V. Ribeiro, A. Seth, S. Ardon, A. Mahanti, and S. Triukose. First Impressions on the State of Cellular Data Connectivity in India. In *Procs. of ACM DEV-4*, ACM DEV-4 '13, 2013.

[13] A. Kvalbein, D. Baltrūnas, J. Xiang, K. R. Evensen, A. Elmokashfi, and S. Ferlin-Oliveira. The Nornet Edge platform for mobile broadband measurements. *Elsevier Computer Networks special issue on Future Internet Testbeds*, 2014.

[14] L. Li, K. Xu, D. Wang, C. Peng, Q. Xiao, and R. Mijumbi. A Measurement Study on TCP Behaviors in HSPA+ Networks on High-speed Rails. In *Proc. of INFOCOM*, 2015.

[15] A. Nikravesh, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh. Mobile Network Performance from User Devices: A Longitudinal, Multidimensional Analysis. In *Procs. of PAM*, 2014.

[16] P. Perala, A. Barbuzzi, G. Boggia, and K. Pentikousis. Theory and Practice of RRC State Transitions in UMTS Networks. In *IEEE GLOBECOM Workshops*, 2009.

[17] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing Radio Resource Allocation for 3G Networks. In *Proc. of IMC*, 2010.

[18] S. Rosen, H. Luo, Q. A. Chen, Z. M. Mao, J. Hui, A. Drake, and K. Lau. Discovering Fine-grained RRC State Dynamics and Performance Impacts in Cellular Networks. In *Proc. of Mobicom*, 2014.

[19] J. P. Rula, V. Navda, F. Bustamante, R. Bhagwan, and S. Guha. "No One-Size Fits All": Towards a principled approach for incentives in mobile crowdsourcing. In *Proc. of IMC*, 2014.

[20] S. Sen, J. Yoon, J. Hare, J. Ormont, and S.Banerjee. Can they hear me now?: A case for a client-assisted approach to monitoring wide-area wireless networks. In *Proc. of IMC*, 2011.

[21] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang. Characterizing and Modeling Internet Traffic Dynamics of Cellular Devices. In *Proc. of SIGMETRICS*, 2011.

[22] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang. A first look at cellular network performance during crowded events. In *Proc. of SIGMETRICS*, 2013.

[23] J. Sommers and P. Barford. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In *Proc. of IMC*, 2012.

[24] F. P. Tso, J. Teng, W. Jia, and D. Xuan. Mobility: A Double-Edged Sword for HSPA Networks. In *Proc. of MobiHoc*, 2010.

[25] Y. Xu, Z. Wang, W. Leong, and B. Leong. An End-to-End Measurement Study of Modern Cellular Data Networks. In *Proc. of PAM*, 2014.

"Infect-me-not": A User-centric and Site-centric Study of web-based malware

Huy Hang*, Adnan Bashir†, Michalis Faloutsos*, Christos Faloutsos‡, and Tudor Dumitras§

* University of California, Riverside
Department of Computer Science and Engineering
Riverside, CA 92521
{hangh, michalis}@cs.ucr.edu

‡ Carnegie Mellon University
Department of Computer Science
Pittsburg, PA 15213
christos@cs.cmu.edu

† University of New Mexico, Albuquerque
Department of Computer Science
Albuquerque, NM 87131
abashir@cs.unm.edu

§ University of Maryland, College Park
ECE Department
College Park, MD 20742
tdumitra@umiacs.umd.edu

Abstract

Malware authors have been using websites to distribute their products as a way to evade spam filters and classic anti-virus engines. Yet there has been relatively little work in modeling the behaviors and temporal properties of websites, as most research focuses on detecting whether a website distributes malware. In this paper we ask: How does web-based malware spread? We conduct an extensive study and follow a website-centric and user-centric point of view. We collect data from four online databases, including Symantec's WINE Project, for a total of more than 600K malicious URLs and over 500K users. First, we find that legitimate but compromised websites constitute 33.1% of the malicious websites in our dataset. In order to conduct this study, we develop a classifier to distinguish between compromised vs. malicious websites with an accuracy of 95.3%, which could be of interest to studies on website profiling. Second, we find that malicious URLs can be surprisingly long-lived, with 10% of malicious sites staying active for three months or more. Third, we observe that a significant number of URLs exhibit the same temporal pattern that suggests a flush-crowd behavior, inflicting most of their damage during the first few days of appearance. Finally, the distribution of the visits to malicious sites per user is skewed, with 1.4% of users visiting more than 10 malicious sites in 8 months. Our study is a first step towards modeling web-based malware propagation as a network-wide phenomenon and enabling researchers to develop realistic assumptions and models.

I. Introduction

Distributing malware indirectly via web-pages has become a very popular way for spreading malware in the last 8 years. In 2012, Google reported that they identify 9,500 of malware-spreading websites each day [1]. These websites infect their visitors, but we can identify two different types: (a) the **born-malicious**, which are registered and operated by the malicious entities, and (b) the **compromised**, legitimate websites infiltrated by hackers and injected with malware.

"How does web-based malware spread?" is the key question that motivates this work. We consider a site-centric and a user-centric point of view: (a) what is the behavior and the lifecycle of the website that spreads malware, and (b) what is the behavior of the users that visit such websites. Our goal is three-fold: (a) investigate the composition of the websites to find out how many of them are born-malicious and how many compromised, (b) understand the life of a malicious URL and its impact, and (c) identify patterns in the way users visit malicious URLs. For the remainder of this paper, we use the term malware to refer to web-based malware. In our work, we focus on the spatiotemporal patterns of how malware is distributed from malicious sites to users.

Most previous work has focused more on identifying malicious websites, and less on their propagation patterns. In more detail, we identify four areas of focus in the literature: (a) the identification of websites vulnerability to infiltration, (b) the detection of websites actively distributing malware, (c) the study of the ecosystem and the techniques used by hackers, and (d) the analysis of the web-based malware themselves. We describe research efforts in these areas in section V.

Our key contribution is an extensive study of user exposure to web-based malware following both a site-centric and user-centric point of view. We use two data sets: (a) D_{ODB}, with roughly 66K malicious URLs collected from four online databases between December 2013 to September 2014, and (b) D_{WINE}, which captures visits to malicious websites from roughly 530K users from January 2011 to August 2011 collected by the Symantec's WINE Project. Note that Symantec's data captures the exposure of the users to malware as seen by its anti-virus products, as we explain in section II.

Our work can be summarized into the following major observations.

a) **Compromised websites play a significant role in malware dissemination.** We find that among all the domains in our D_{ODB} dataset, 33.1% of them belong to compromised websites. For our study, we developed a

ISBN 978-3-901882-83-8

Machine Learning-based method to distinguish compromised websites from born-malicious sites. Our approach exhibits a 95.3% accuracy. We want to stress that the ML method we developed is strictly for a forensics purpose: we want to measure how prevalent the phenomenon of compromised websites is and raise awareness about the danger that they may pose. We did not intend the method to be a detection tool for compromised sites.

b) **A malicious URL often distributes many different malware binaries but each malicious binary is typically distributed by one URL.** We find that 33% of the URLs with at least 5 visits in D_{WINE} distribute two or more different binaries (different MD5 hash values). This percentage increases to 46% among all websites with more than 20 visitors. These website are either: (a) distributing completely different malware, (b) using polymorphism to distribute the mutated versions of the same malware to escape detection. In contrast, most malicious binaries (94.6%) are distributed by one URL in our data set.

c) **Most malicious URLs are short-lived, but 10% of them are active for more than three months.** Although 71.6% of URLs in D_{WINE} appear for only one day during the 8 months, roughly 10% stay active for at least three months and a much smaller number have been active for four years. This suggests there may not be an efficient technical and/or legal process to clean up or take down a malicious website.

d) **The "Space-needle" pattern: Many URLs exhibit the same bursty temporal pattern aligned with a campaign-like behavior.** Here, we focus on Highly Active URLs in D_{WINE}, which have lifespans of at least 30 days and have at least 100 visitors each. We find that the time series of the visits to 45.6% of those URLs follow a bursty pattern, which we refer to as "space-needle" due to its shape. URLs following this pattern usually peak within the first two days of their life, and the maximum number of daily visits is at least an order of magnitude larger than the median, as we discuss in section IV.

e) **The distribution of the visits to malicious sites per user is skewed and can be described by a power law of exponent $-\frac{1}{2}$.** A small percentage of users in D_{WINE} are highly susceptible to visiting malicious URLs. For example, we find that 1.4% of all users (close to 7500 users!) in our data set visited at least 10 malicious URLs during the 8 months.

Data Archive and Acknowledgment. The data is available for follow-up research as reference data set WINE-2014-002 in Symantec's WINE repository. We are grateful to Drs. Matthew Elder and Daniel Marino of Symantec Research Lab for their support and feedback.

II. Our Data Sets and Background

We present the sources of information and data sets that we use in our work.

A. Sources for URL characterization

We rely mostly on two sources of information regarding the status of a URL. First, VirusTotal is a popular online service where a user can submit a binary or a URL or a domain so it can be scanned by at least 50 anti-virus engines. Once the scan concludes, the user can retrieve a report that shows, in the case of a domain, the number of AV engines that considered the domain to be of a malicious website. We call this value the VirusTotal Malicious Score of a domain. Second, the Web of Trust (WoT) Reputation Score is a numerical value between 0 and 100, inclusively, given to a website by WoT [2], which relies on its user community to rate the websites the users came across. The higher score a website has, the more trustworthy. A poor reputation score does not imply a website is malicious.

B. Our data sets

We use the following sources to build our data sets.

1) Online databases: We collected malicious URLs from four different online databases: Cybercrime Tracker [3], Malc0de [4], Malware Domain List [5], and VX Vault [6]. These online databases are maintained by communities and publish new malicious URLs on a regular basis. We began collecting the URLs in March 2014 and continued to do so every day until September 2014. This data set, which we call D_{ODB} from this point onward, will be used to build a classifier to distinguish born-malicious from compromised websites.

	URLs	Domains	Clients	MD5s
O.D.B.	71,542	8,724	-	-
WINE	626,472	106,026	530,061	504,324

TABLE I
Data from Online Databases (O.D.B.) and WINE

This dataset is used exclusively to train and test our classifier of born-malicious and compromised websites, as will be shown in section III.

2) Symantec's WINE data: Symantec's Worldwide Intelligence Network Environment (WINE) [7] is a massive corpus of telemetry data sampled from more than 120 million machines, both enterprise and consumer, and made available to the research community. This dataset was also used in the analysis of zero-day attacks in [8] as well as the study to expose the change in cyber threat landscape and the emergence of new attack surfaces [9].

The WINE database is divided into five datasets, each containing data from different aspects of the data collection process. The data that we collected from WINE belong to two specific datasets:

1) AV Telemetry: data collected from all clients any time a Symantec AV product detected that a *malicious* binary **executable** was downloaded.
2) Binary Reputation: data collected on binary **executables** downloaded by users in Symantec's reputation-based security program. Even though this dataset contains information on both malicious and benign binaries, it does

not contain information on whether a specific binary is malicious.

We use the **machine ID identifier** to distinguish users. This is a unique ID that each Symantec software installation. This way, we eliminate the "noise" that can be introduced by using IP addresses, which are dynamically assigned and often obfuscated by Network Address Translators (NATs).

C. Modeling the exposure to malware

The WINE data we collected focuses exclusively on *visits to malicious URLs*. Every entry in the dataset represents a report any time a user **downloaded** a malicious binary from a URL. However, we do not make a claim as to whether the user was infected or not. In fact, we believe that the malicious binaries were detected and the users would have been protected, unless they explicitly overrode the antivirus warning.

Each data point in our data set, which we will call D_{WINE}, contains: (a) the timestamp of the receipt of the report at a Symantec server, (b) the URL from which the binary was downloaded, (c) the MD5 hash of the binary, and (d) the `ID` of the client machine.

We begin with collecting the information about malicious URLs from AVs Telemetry from January to August 2011. We then correlate with Binary Reputation to obtain the information about the URLs from *before* Symantec determines that the URLs were distributing malware so that we get the complete history about each URL (including which binaries they distributed and who downloaded from them).

We use this dataset to conduct analysis of the spatiotemporal characteristics and malicious websites and study the behaviors of the users who visit the malicious sites, the details of which will be shown in section IV.

Representativeness. The classic question for any real measurement data is how representative is the data. We rely on users that have Symantec AV products installed, and this may be introducing some bias, though we don't have any reason to believe that users of other anti-virus solutions will have an i ntrinsically different behavior. At the same time, D_{WINE} is data collected from roughly 550K users spanning eight months and WINE is drawn from more than 120 million machines worldwide, so our dataset consist of a reasonably wide cross-section of users.

III. Provenance of malicious websites

Given a malicious website, we would like to determine if it is born-malicious or compromised (which we defined in the introduction). We present our Machine Learning-based method that accomplishes this with high accuracy. Note that we use the dataset D_{ODB} exclusively in building the classifier and performing testing.

Why do we want to study malware-spreading websites, after they have been identified as such? There are two reasons. First, compromised websites are not very well studied, to the best of our knowledge despite the fact that alarms had been raised about them. In their 2014 Threat Report [10], Symantec discovered that one in eight legitimate websites have unpatched critical vulnerabilities, making them ripe for an attack. Second, hackers try to infect their victims by abusing the *trust* that legitimate sites have established over time instead of getting around domain or IP blacklists by creating new websites or constantly switching to new IPs via fast-fluxing.

The proposed technique is arguably the first that focuses on this problem. As such, we would offer it as a publicly-available tool for studying the provenance of websites (whether they are created for malicious purposes or hijacked) and providing a first-level forensics capability. Note that the course of action for stopping the spread of malware depends on this classificiation. In the case of a born-malicious site, the site needs to be taken down and possibly have the hosting entity notified.

A. Building the classifier.

We present the steps that we took for developing our classifier.

1) Data Preprocessing: Even though the D_{ODB} dataset includes 8,724 domains in total, we run the classifier on only 3,975 of them. We do not include in our study the domains that belonged to any of the following categories:

a) Domains not resolving to IPs. These 1,550 domains no longer provide valid DNS records, because there was a time gap between when the domain was reported as malicious and when we attempted to crawl them. A close examination of such domains shows most have poor reputation scores and created recently. It is likely these domains were deactivated for distributing malware.
b) Domains returning `40X` codes or no content.
c) Domains belonging to known Content Delivery Networks, file-sharing sites (e.g. `mediafire`) and websites hosting free software (e.g. `softpedia`). By nature, these websites allow the posting of user content, which can often point to malicious websites or even contain malware.

2) Our training and testing data: From the remaining 3,975 domains (which we call D_{classify}), we randomly selected 609 domains and split them into two used for training and then testing our binary classifier:

a) $D_{\textbf{train}}$ has 200 domains, 139 labeled as compromised and 61 born-malicious
b) $D_{\textbf{test}}$ has 409 domains, 280 labeled compromised and 129 malicious.

We label the domains manually and carefully: we visit each domain in a browser in a virtual machine, carefully examine each landing page we come across, and explore each link on the landing page to see if we can reach other pages that may contain legitimate content. The richer the content is, the more confident we are that the website is a legitimate website that was compromised.

3) Features: We first present the features that our classifier uses, and later we discuss other features that we considered but did not ended up using. We use the following features:

a) Number of URLs embedded in the landing page.

b) Number of images on the landing page.
c) Age (days) of domain since registration.
d) Web of Trust's reputation score for domain.
e) VirusTotal's malicious score for domain.

Intuitively, a legitimate but compromised website tends to bear the following characteristics:

a) Its landing page is much more complex than that of a website created to deliver malware, meaning that it has richer content, more hyperlinks that lead to its other pages, and more images in general.
b) It has been around for longer than a born-malicious website, as malware authors stage new websites very often, knowing that it is likely that a malicious site could be taken down or blacklisted quickly. The older a website is, the more trustworthy it is.
c) Compromised websites have relatively higher Web of Trust's score.

These aforementioned characteristics are covered by features (1), (2), and (3), which are not enough, as using only these three means that we run the risk of classifying a newer, simple website as malicious or classifying an older and more complex malicious site as benign. To avoid this, we also make use of features (4) and (5), which allows the classifier to factor into its classification decision how many anti-virus engines consider the site to be malicious and how good a reputation a site may have.

To obtain relevant statistics for each domain in D_{ODB}, we created an automatic web crawler using Selenium [11] and connected the Selenium-driven browser to a proxy to keep track of every image downloaded by the browser.

4) Other features: We have also considered other features to use in our classifier such as: the total number of pages hosted by a website, the total number of images and links that could be found on all of the pages, etc. We opted to not use these features. First, we want to create a light-weight classifier, so we avoid features that intense in computation and resource utilization. Using a feature such as the total number of pages on a website would mean that our crawler would have to crawl the entire website and explore every single link that can be discovered while making sure that the crawler would not follow a link that leads out of the website. Further, as we show below, we were able to achieve good classification accuracy using the selected light-weight features.

We also considered using IP blacklists as an additional means of pre-filtering to quickly identify a malicious website. We also decided against using the black-lists because a single IP address may be home to multiple websites and we run the risk of labeling as malicious a benign website hosted on the same server.

B. Training the classifier.

We use D_{train} to train our classifier, and we select the Random Tree method, because it gives the best performance among all others included in the WEKA Machine Learning framework [12]. We tried to create single-feature classifiers to test the accuracy of each feature but **none** of the classifiers exceeded 85% in accuracy (as seen in Table II) when applied on D_{train}, where we define accuracy as the ratio of the number of correctly labeled domains and the total number of domains. Applying a classifier built from all features on D_{train} yields no misclassified instances. Note that we define accuracy as the number of correctly labeled instances over the total number of instances.

Feature name	Accuracy
Number of URLs on landing page	82.3%
Number of images on landing page	83.7%
Age since registration	84.7%
Web of Trust score	77.4%
Virus Total score	73.5%

TABLE II
ACCURACY USING A CLASSIFIER WITH ONLY ONE FEATURE

We also performed **cross-validation** of our training set using the 10-fold cross validation function of the WEKA suite, and the result (using all five features) is just as good in that there is no instance misclassified.

C. Testing the classifier.

For testing, we use the D_{test} dataset.

1) Achieving a classification accuracy of 95.3%: We find that the number of correctly classified instances is 390 out of 409. We investigated the misclassifications to understand the limitations of our approach. Among the 19 misclassified domains, we find: 9 compromised that were labeled as born-malicious and 10 malicious domains that were labeled as compromised. Our careful investigation shows that some misclassifications happened due to several reasons: (a) the domains were hosted by dynamic DNS services, so the age values reported, which are very high, are of the DNS services themselves, (b) some compromised websites have extremely simple home pages with few images and embedded URLs.

2) 33.1% of malicious domains are compromised: With our classifier, we classify all the domains in D_{classify}. We find 2,885 compromised and 1,090 born-malicious. This means that roughly 33.1% of the domains from D_{ODB} are benign websites infiltrated by hackers and used to distribute malware. We will revisit the phenomenon of compromised domains again in the next section.

IV. PROFILING MALICIOUS URLS & THEIR VISITORS

We present our findings on the temporal properties of malicious URLs and the browsing behavior of the users.

A. Malicious URLs distribute many different malware binaries but each binaries is usually distributed from one URL.

In D_{WINE}, we observed many instances where the same URL yielded binaries with different MD5 hashes, often within the same hour. This phenomenon can be observed in Figure 1, where the each data point represents a single binary executable (X-axis) and the number of distinct URLs (Y-axis) from which it can be downloaded.

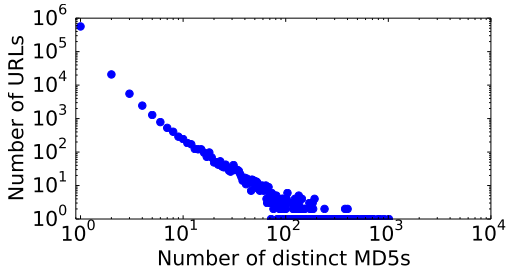


Fig. 1. Distribution of the number of unique MD5 hashes seen per URL. For example, the datapoint (x, y) indicates that there are x binaries each of which can be downloaded from exactly y URLs.

In this paper, we will call this phenomenon *URL-centric polymorphism*. Note that malware polymorphism, in general, refers to distributing the same malware in many different versions. We choose a different name to stress that we only focus on MD5-hash-based similarity: we were not given access to the malware itself to determine if two MD5 hashes correspond to the same malware.

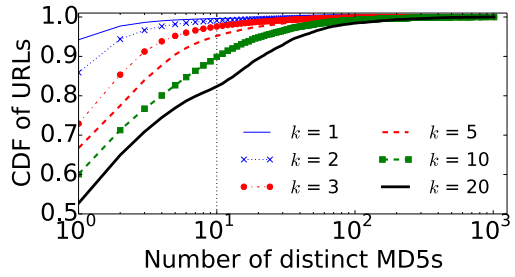


Fig. 2. URL-centric polymorphism observed more clearly on highly-active URLs.

1) One website many MD5s: In Figure 2, we show the distribution of unique MD5 hashes for each URL. Each curve, represented by a k value, shows the distribution of MD5 count per URL where the URLs have at least k visitors. We can see for 94% of URLs, each is associated with only one MD5 ($k = 1$). Most of these low-access URLs have one or two visitors, which makes it difficult to observe MD5 variations. The polymorphism becomes more evident for URLs with more visitors. For the URLs with at least $k = 5$ visitors, 33% of them distributed more than one binary, but this number increases to 46% when for URLs with at least $k = 20$ visitors.

2) Each MD5 is typically distributed from one website.: Reversing the question, we examine how many websites distribute the same MD5 malicious binary in D_{WINE}. Towards this goal, we look at each MD5 hash value in our D_{WINE} dataset and count the number of distinct malicious URLs from where the binary with the MD5 was observed to be downloaded. In Figure 3, we plot the distribution of MD5s according to the number of URLs that distribute them. We observe that 92.2% (more than 464K binary executables) are distributed by only one single URL.

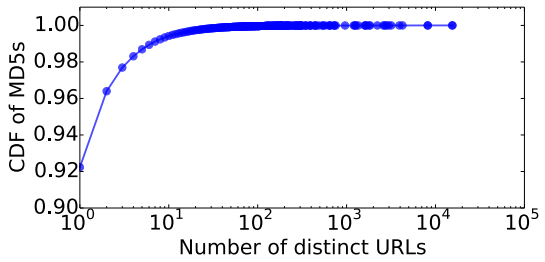


Fig. 3. Distribution of the number MD5s that are distributed by a certain number of distinct URLs (x-axis).

3) A few MD5s are widely distributed by more than 100 URLs.: There are some binary executables that appeared on more than a hundred URLs, which we did not expect. To investigate, we randomly picked 600 of these binaries and examined the related URLs. We observed:

1) The majority of these binaries (76.2%) appeared on multiple born-malicious domains, which seemed "disposable": the domains typically had random sequences of characters, pointing to an *automated* name-generation process.
2) A relatively small percentage (14.5%) of these binaries appeared on multiple domains that belong to popular file-sharing websites or well-known software distributors. This means malware authors rely on the many free file-sharing services or embed malicious code into popular and often pirated software to distribute the files. Note that although we filtered out such domains from D_{ODB} (as noted in section III-A1, we did not do so for D_{WINE} because our goal is to study how malicious binaries are distributed at large.
3) 9.3% appeared on what seem like compromised sites, many of them active and containing legitimate content. These binaries were distributed by URLs that have similar structure. For example, one such binary was distributed by URLs of the form: `http://{D}/images/facebook-pic-{X}.exe`, where `{D}` represents different domain names and `{X}` a sequence of random digits. This suggests that the sites were compromised: (a) through the use of the same hacking toolkit, and/or (b) by the same hacker.

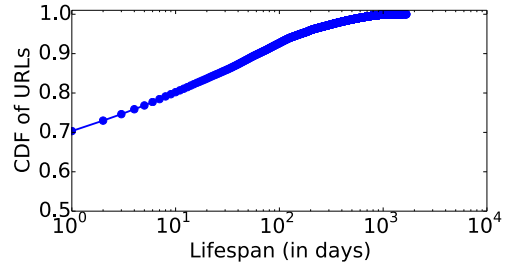


Fig. 4. Cumulative distribution of URLs according to their lifespan (x-axis)

B. Malicious URLs exihibit short lifespans and the number of visitors who visit them follow a skewed distribution

1) Most malicious URLs have short lifespan, but a small percentage live for more than three months: In Figure 4, we study the distribution of the lifespan of websites. We find that 70.6% of all malicious URLs are what we call *single-day URLs* as they appear for only one day in our dataset. Surprisingly, 10% of these websites managed to stay "alive" and actively distributed malware for more than three months and there are 194 malicious URLs that were around for four years. Furthermore, a small percentage (2,427 URLs, making up 0.4% of all URLs) attracted at least a hundred users during their lifespan.

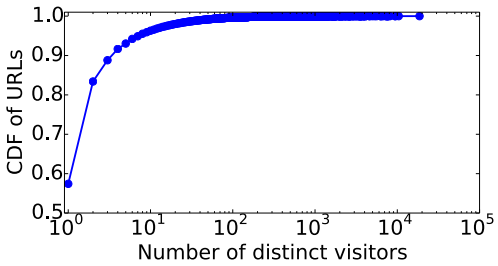


Fig. 5. Cumulative distribution of URLs according to their visitor count (x-axis)

2) The number of visitors per URL follows a skewed distribution: In Figure 5, we plot the distribution of unique visitors per URL. We find that this distribution is highly skewed with 57.4% of the URLs having one visitor, while 11.2% have least three visitors. It can be observed from the Figure that there are malicious URLs whose visitor counts exceed one thousand visitor apiece. We then investigated the top five URLs with the highest number of visitors and observed that they are URLs whose first appearances date back as far as 2010, individually spanning at least a year and a half. One such URL, for example, was distributing a screen-saver program that was flagged by Symantec as distributing Trojan viruses. This observation (a) explains why they managed to attract such a high number of visitors and (b) underscores the fact that even though they have been distributing malicious content, they were never shut down in a timely fashion.

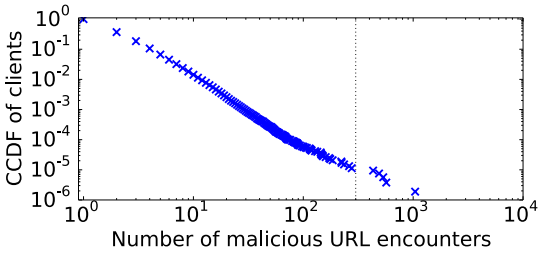


Fig. 6. CCDF of clients with respect to malicious URLs encounter

3) The distribution of the visits to malicious sites per user is skewed and can be described by a power law of exponent -1/2: In Figure 6, we plot the CCDF of the number of visits to malicious sites for each user. We find that the distribution of the number of malicious URL encounter per person seems to follow a power law distribution with exponent $\alpha = -\frac{1}{2}$. Thus, the good news is that most users in D_{WINE} encounter malicious URLs very infrequently. In Figure 7, we plot the CDF of the same distribution and show that 63% of all users visited a malicious URL only once during the entire eight months. However, 1.4% (roughly **7,500** users) visited malicious URLs at least ten times during the same amount of time. This in general suggests that a small group of users were far less cautious than others in their browsing activities.

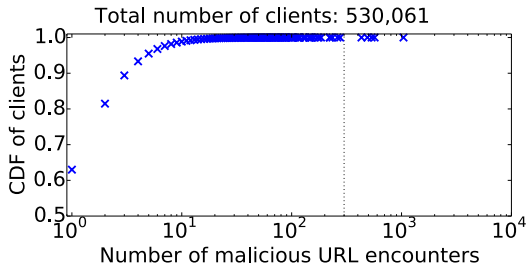


Fig. 7. CDF version of Figure 6

4) Outliers: users with more than 400 visits to malicious URLs: In both Figures 6 and 7, we see that a few data points are well separated from the rest of the distribution (to the left of the dotted vertical lines). Each of these points represents users who visited roughly 400 *distinct* malicious URLs during the eight-month period, averaging to at least *two* malicious URLs a day. We want to stress that each time one of these users visited one such malicious URL, a malicious file was downloaded by their browser and blocked from execution by the Symantec anti-virus product and the user would be subsequently notified. The most active user in our dataset visited a total of 1042 distinct malicious URLs for a duration of 242 days, averaging at least 4 a day. This behavior seems unlikely for a human, so we rule out this possibility.

Upon further investigation, we arrived at two possible explanations for these outlier points.

(i) These behaviors were generated by automated programs, for example a crawler whose purpose is to measure the uptime or downtime of a website or to seek out malicious domains. These programs could have been deployed by researchers.

(ii) Recall that each user in the D_{WINE} dataset is identified by a unique Machine ID, which is given by the Symantec software (think product number). It is possible for a user to install the AV product in a Virtual Machine and clone it, thereby allowing multiple Virtual Machines to report their activities to Symantec with the same Machine ID.

C. Space-needle: Many highly active URLs exhibit the same bursty temporal pattern that suggests a campaign-like behavior.

We discover that the visits to many URLs exhibit a bursty behavior, as can be seen in Figure 8. This temporal pattern,

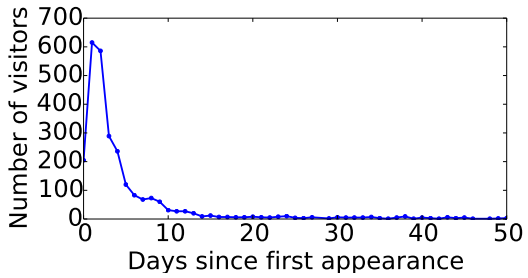


Fig. 8. One example of the space-needle propagation pattern

which we will call "space-needle", could be the result of an active campaign, staged by the hacker, to drive traffic to a newly infected or created site. Consequently, most of their visits take place during the first few days when the site appears, since after the first few days, the spam filters and black lists catch up and reduce the number of visitors.

We want to study the extent of the "space-needle" phenomenon in more detail. We start with focusing on URLs with a lifespan of 30 days or more and at least 100 visitors. Having at least 100 visitors *alone* does not qualify a URL to be highly active, a URL may just have that many visitors on the first few days since its appearance and is taken down afterwards. What we would like to study is, after all, the interesting malicious URLs that are both *long-lived* and had attracted substantial amounts of visitors.

We identify 2,402 URLs in D_{WINE} that meet these criteria. We will refer to these URLs from this point on as **Highly Active URLs**. We then do the following analysis to jointly define the "space-needle" pattern and quantify its presence with a technique that is commonly used in data mining and the steps of which are described below.

Given the set U of Highly Active URLs mentioned above, we begin with the following preprocessing steps with each $u \in U$:

1) We represent the user-visit pattern of each u during the first thirty days of its lifespan with the ordered sequence $V_u = \{(i, v_i^u)\}$ where i is the i^{th} day since u's first appearance and v_u^i is the number of distinct users who visited u on that same day.
   We only preserve the user visits during the first thirty days because (i) they are sufficient to capture most of the user visits to the URLs and (ii) we need to make sure that the activities that were captured from each URL span a uniform amount of time for the purpose of comparison.
2) For each V_u, we proceed to create the time series T_u by using linear interpolation to fill in any existing "gap" (which can be any day that there is no recorded user visit to the URL).

Once we have $T = \{T_u \quad \forall u \in U\}$, we:

1) Select a representative time series T_r that intuitively captures the essence of the "space-needle" (seen in Figure 8) shape and remove it from T
2) Compute the Euclidean distance between each $T_u \in \{T - T_r\}$ to T_r.
3) Sort each $T_u \in \{T - T_r\}$ so that if T_u precedes $T_{u'}$, $E(T_u, T_r) \leq E(T_{u'}, T_r)$ where E denotes the Euclidean distance function.
4) Manually inspect the "shape" of each time series from the beginning of the sorted list until we come across a time series that no longer visually resembles that of the representative time series T_r.

At the end of this process, we find 1,095 URLs or 45.6% of the 2,402 highly-active URLs, which we will call **Space-Needle URLs**.

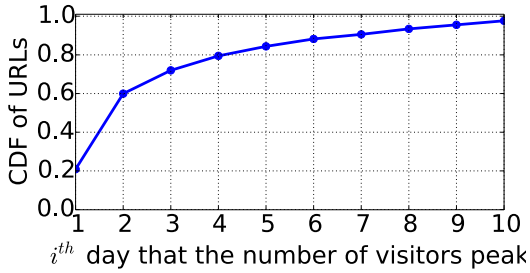


Fig. 9. Distribution of which day the URLs gained the maximum number of visitors

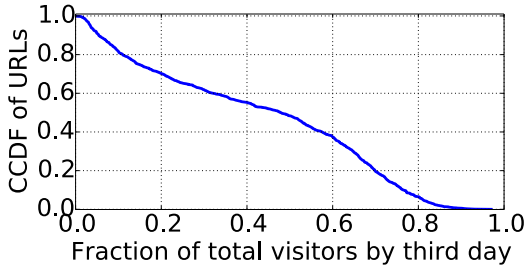


Fig. 10. Distribution of the fraction of total number of visitors each URL accumulated by the third day

The next step is to quantify the properties of the Space-Needle URLs. We can see from Figures 9 and 10 that 60% of these URLs achieved their peak number of daily visitors either on their first day of appearance, or the very next day. By the end of the third day, 50% of all the Space-Needle URLs have seen at least half of their total number of visitors. Moreover, for 80% of the Space-Needle URLs, the peak number of daily visitors is at least one order of magnitude larger than the median value of daily visitors. Note that we only consider days with at least one visitor to compute the value of the median.

D. Where do malicious domains end up?

When we performed DNS queries on all of the domains of the malicious URLs reported in D_{WINE}, we found that roughly a third of the malicious domains (35.9%) continue to be active (by which we mean that doing DNS queries on them yield IP addresses). We were intrigued as to why these domains (presumably of malicious websites) are still active even though they were first reported in 2011. To further investigate, we

randomly selected 600 still-active domains and accessed them in a browser in a virtual machine and we identified the following categories.

1) Recovered: 36.7% of the domains seem to be benign, bearing legitimate content. We believe they might have been compromised when Symantec detected malicious binaries being distributed by their servers. This suggests that compromised websites seem to have also played a significant role in malware delivery in 2011.
2) File and content sharing: 23.0% of the domains are file-sharing websites and software distributors. This suggests that malware had been uploaded to these sites and long since removed.
3) Parked: 20.2% of the domains are now under control of domain parkers and serving as advertisement space.
4) Not accessible: 20.1% of the domains were not accessible when we tried them, e.g. they returned `40X` error codes or blank pages.

E. Some users are more prone to careless surfing behavior.

We tried to estimate the probability of a user visiting a malicious URL given their history by executing the following steps.

1) Select one month from January to *July* of 2011.
2) Calculate how many URLs each client encountered during that month.
3) Let C_x^i be the set of clients who visit x URLs during month i, and let V^{i+1} be the set of the visitors to malicious URLs during the following month. We compute the percentage of users in C_x^i who will be **repeat offenders**: $P_x^{i \to i+1} = |C_x^i \cap V^{i+1}|/|C_x^i|$.
4) Repeat the steps above for every other month.
5) Compute the *average* $P_x^{i \to i+1} \quad \forall x$ across all months.

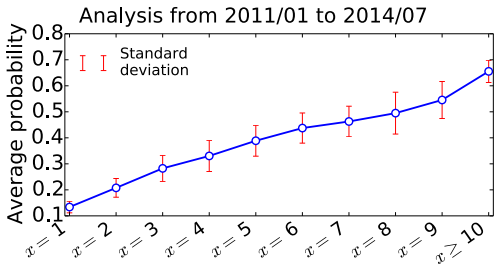


Fig. 11. Probability of a user visiting a malicious URL given the number of malicious URLs visited during previous month

In Figure 11, we plot the average probability of a user visiting a malicious URL within a month given the number of such visits (x) the month before. The average probability is computed across all users with the same number of visits to malicious sites for all pairs of consecutive months. We observe significant increase in the average probability as the number of URLs visited grows, indicating that there are indeed users who are prone to careless surfing behaviors even though they have anti-virus products installed on their computers.

We noted above in section IV-B4 that there are cases where a single machine ID visited thousands of sites. We do not believe that these outliers contribute meaningfully to the phenomenon we described in Figure 11, as there are very few outlier machine IDs and there are more than half a million machine IDs that are observed for this part of the study.

V. Related Work

The aim of our work is different from that of the majority of URL classification methods [13] [14], which focus on distinguishing malicious URLs from non-malicious ones while we focus on identifying whether a site identified as malicious are born-malicious or in fact compromised by hackers.

The most related work to profiling the behaviors of binary distribution by Papelaxakis et al. [15] focuses on benign binaries and presents a model called SHARKFIN that describes the propagation pattern of popular software. A recent work by Kuhrer et. al. [16] evaluates the completeness of black lists and presents a method to identify parked domains and sink holes. In recent work, Li et. al. [17] describe a method to identify a website that is compromised by re-direct script injection. Although this is relevant to our work, the proposed method targets a very specific type of compromised websites, while we need a general method to distinguish compromised websites at large from born-malicious ones.

Overall, there are four areas that touch on various aspects of web-based malware study.

A. Investigating the landscape of web-based malware distribution.

This first area then is split into two smaller ones: (a) how to actively seek out new malicious sites [18][19][20] and (b) the detection of malicious URLs [21][22], drive-by-downloads website [14][23], or malware-infected machines [13]. In [14], the authors statically analyze the content of websites to accomplish the goal of detection of drive-by-download sites and in [23], the authors attempt to detect when a user is redirected multiple times and eventually delivered to a website managed by malware distribution networks by analyzing the URLs in the redirect chains themselves.

In [13], Invernizzi et. al. invest their effort into the detection of machines in large-scale networks that meet the following criteria: (i) the machines have already been infected by drive-by-download attacks and (ii) the small piece of code dropped into each machines is sending HTTP requests to remote hosts to down- load the full payload for the installation of the malware. This work is unlike ours in that the goal of the work is to identify the infected hosts and, consequently, can be used for identify malicious websites that provide the malware payload. The author, however, never put a focus on identifying compromised websites.

B. Detecting vulnerable sites.

This area focuses more on the identification of websites that may be at risk of infiltration [24][25]. In this work [25], the authors created a classifier that identifies websites that may

become compromised in the future by automatically extracting content-based features from a large corpus of labeled websites as well as using out-of-band information such as Alexa ranking the Amazon Web Information Service. In [24], Canali et. al. hosted vulnerable websites on many hosting providers and tried to compromise the sites themselves. They showed, alarmingly, afterwards that the providers are unable to detect simple signs of malicious activities.

C. Studying the malware ecosystem.

This area studies the ecosystem that supports the malware distribution, providing insights on the attacks carried by malicious websites on the users [26] or on the infrastructure that supports malware authors [14][27], enabling them to spread their malicious software for monetary gains.

D. Malware binary analysis and classification.

This area focuses on the analysis of the web-based malware binaries [28][29][30]. In [28], the authors extracted features from the HTTP traffic traces generated by the malware installed on safe environments and used those features (which included total number of requests, average number of parameters, etc.) to cluster the malware samples. From the clusters, signatures can be generated to detect when a computer may be infected. Rossow et. al. monitored more than 100,000 malware samples at runtime in their Sandnet environment [29] and observed their network behaviors, thereby showing that DNS and HTTP are the two protocols most common among those used by the malware. In [30], Rossow et. al. extended their work to 23 different malware downloaders, most of which were yet documented. The authors characterized them according to their communication models, investigated their resilience, and analyzed how they they used DNS and fast-flux techniques to carry out their operations.

The work in this fourth area, while dealing directly with network-based malware, is of little help to us as they are malware-centric, as never got access to the binaries.

VI. Conclusion

In this paper, we focus on modeling the user exposure to web-based malware by analyzing more than 500K users accessing roughly 600K URLs from a data set collected from Symantec's WINE Project. We find that:

a) Compromised websites play a significant role in malware dissemination, as 33.1% of them in D_{ODB} dataset are compromised websites. In section III, we showed the different ways in which a compromised websites are fundamentally different from a born-malicious one.

b) A malicious URL often distributes many different malware binaries but each malicious binary is typically distributed by one URL.

c) Most malicious URLs (71.6%) are short-lived, but 10% of them are active for more than three months in D_{WINE}.

d) The number of visitors of many malicious website exhibit a bursty campaign-like temporal pattern, which we refer to as the "Space-needle" pattern.

e) The distribution of the visits to malicious sites per user is skewed and can be described by a power law of exponent $-\frac{1}{2}$.

Our study is a first step towards modeling web-based malware exposure and could help us understand malware distribution as a network-wide phenomenon.

References

[1] E. Mills, "Google finds 9,500 new malicious Web sites a day," www.cnet.com/news/google-finds-9500-new-malicious-web-sites-a-day/, June 2012.

[2] "WoT Reputation API," https://www.mywot.com/wiki/API.

[3] "Cybercrime Tracker," http://cybercrime-tracker.net/.

[4] "Malc0de Database," http://malc0de.com/database/.

[5] "MDL," http://www.malwaredomainlist.com/.

[6] "Vx vault," http://vxvault.siri-urz.net/ViriList.php.

[7] T. Dumitras and D. Shou, "Toward a standard benchmark for computer security research: The worldwide intelligence network environment (wine)," in *BADGERS 2011*. ACM.

[8] L. Bilge and T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in *CCS 2012*. ACM, pp. 833–844.

[9] K. Nayak, D. Marino, P. Efstathopoulos, and T. Dumitras, "Some Vulnerabilities Are Different Than Others: Studying Vulnerabilities and Attack Surfaces in the Wild ," *RAID 2014*.

[10] "Symantec's 2014 Security Threat Report," http://www.symantec.com/security_response/publications/threatreport.jsp.

[11] "Selenium, Web Browser Automation," http://www.seleniumhq.org/.

[12] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham, "WEKA: Practical machine learning tools and techniques with Java implementations," 1999.

[13] L. Invernizzi, S.-J. Lee, S. Miskovic, M. Mellia, R. Torres, C. Kruegel, S. Saha, and G. Vigna, "Nazca: Detecting malware distribution in large-scale networks," in *NDSS 2014*.

[14] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: a fast filter for the large-scale detection of malicious web pages," in *WWW 2014*. ACM.

[15] E. E. Papalexakis, T. Dumitras, D. H. P. Chau, B. A. Prakash, and C. Faloutsos, "Spatio-temporal mining of software adoption & penetration," in *ASONAM 2013*. ACM.

[16] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *RAID 2014*.

[17] Z. Li, S. Alrwais, X. Wang, and E. Alowaisheq, "Hunting the red fox online: Understanding and detection of mass redirect-script injections," in *S&P 2014*.

[18] Z. Li, S. Alrwais, Y. Xie, F. Yu, and X. Wang, "Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures," in *S&P 2013*. IEEE.

[19] C. Seifert, I. Welch, P. Komisarczuk, C. U. Aval, and B. Endicott-Popovsky, "Identification of malicious web pages through analysis of underlying dns and web server relationships." in *LCN 2008*.

[20] J. W. Stokes, R. Andersen, C. Seifert, and K. Chellapilla, "Webcop: Locating neighborhoods of malware on the web," in *LEET 2010*.

[21] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: Url names say it all," in *INFOCOM 2011*. IEEE.

[22] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *SIGKDD 2009*.

[23] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee, "Arrow: Generating signatures to detect drive-by downloads," in *WWW 2011*.

[24] D. Canali, D. Balzarotti, and A. Francillon, "The role of web hosting providers in detecting compromised websites," in *WWW 2013*.

[25] K. Soska and N. Christin, "Automatically detecting vulnerable websites before they turn malicious," in *Usenix Security 2014*.

[26] N. P. P. Mavrommatis and M. A. R. F. Monrose, "All your iframes point to us," in *Usenix Security 2008*.

[27] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis *et al.*, "Manufacturing compromise: the emergence of exploit-as-a-service," in *CCS 2012*. ACM.

[28] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces." in *NSDI 2010*.

[29] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. Van Steen, F. C. Freiling, and N. Pohlmann, "Sandnet: Network traffic analysis of malicious software," in *Proceedings of the Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. ACM, 2011, pp. 78–88.

[30] C. Rossow, C. Dietrich, and H. Bos, "Large-scale analysis of malware downloaders," *DIMVA 2013*.

Securing the Private Realm Gateway

Hammad Kabir, Jesús Llorente Santos, Raimo Kantola
Department of Communications and Networking
Aalto University
Helsinki, Finland
{hammad.kabir, jesus.llorente.santos, raimo.kantola}@aalto.fi

***Abstract*—The traditional mechanisms to traverse Network Address Translators (NAT) do not scale well to battery powered mobile-hosts: the majority of Internet users today. Private Realm Gateway (PRGW) aims to replace NATs at network edges and overcome the drawbacks of the NAT traversal mechanisms. The solution does not require changes in end-hosts or protocols, and hosts in the private realm can remain globally reachable without polling. PRGW handles incoming connections based on domain resolution of the served hosts. Incoming DNS queries create connection state in PRGW for subsequent packet forwarding. The connection state provides means for access control on the Internet-originated flows. This paper analyses the security of PRGW and introduces mechanisms that protect the served hosts and networks against Internet-borne attacks, in particular: address spoofing and Distributed Denial of Service (DDoS). The paper contributes to establish PRGW as an incrementally deployable network function that offers light-weight NAT traversal and protects the private realm against the inherent Internet threats.**

Keywords— Security; Gateway; NAT Traversal; PRGW; DNS; NAT; Denial of Service; DDoS; Internet threats; Network;

I. INTRODUCTION

According to ITU-T, mobile broadband subscriptions have reached 3.2 billion individuals connected to the Internet [1]. This growing number of mobile users raises challenges for the Internet and further aggravates the IPv4 address space depletion problem. The adoption of NAT at network edges alleviated the IPv4 address space exhaustion at the cost of introducing the reachability problem, which prevents the Internet hosts from unilaterally initiating a connection to hosts in the private realm. The mobile hosts typically reside in the private address space; however the IETF recommended methods for NAT traversal [2] scale poorly to battery-powered hosts [3] and communication applications: 1) device has to periodically wake-up to keep its NAT binding alive; and 2) session setup requires exchanging hundreds of overhead messages per application that seeks global reachability, leading to extra power consumption on the device and delays in the session setup.

In [4], we address these drawbacks of the classical NAT traversal mechanisms and propose the Private Realm Gateway (PRGW) solution. The solution does not require any changes in end-hosts, i.e. clients and servers in the private network can stay globally reachable without applications having to run the code for NAT traversal or to maintain their NAT binding. PRGW can be deployed either as a standalone replacement of NATs or as a component of a customer edge switching [5-6] node, at the network edge.

However, as PRGW makes end hosts reachable in the private realm, it will open new opportunities for the hackers to target the private hosts and their network. The increasing reliance of users on their smart phones and mobile apps have presented mobile networks and their hosts as lucrative targets to Internet hackers. As a result, they are subject to a wide variety of threats possible in the Internet.

The paradigm of Internet security can be viewed as an arms race between attackers and defenders. The possibility of source address spoofing, distributed denial of service (DDoS), traffic floods and network/port scans is inherent in the Internet. Today, hackers often abuse free services, e.g. Google DNS, and employ compromised hosts as reflectors/amplifiers in launching their attacks. The outcome of these attacks may lead to excessive network usage, computing downtime, service unavailability, and ultimately waste of human capital [7]. Societies heavily rely on the Internet, and use it for mission-critical activities. Therefore, the networks shall deploy mechanisms that protect their hosts and resources against Internet-borne attacks, in particular source address spoofing, network scans and DoS, which are often used as launch point for more advanced attacks. Consequently, our threat model in the paper spans to the above attack types.

In this paper, we seek to provide mechanisms that protect PRGW and make it a feasible function in modern IP-networks. As a result, PRGW emerges as a network function that besides overcoming the drawbacks of the NAT traversal solutions [4] is hardened against the inherent Internet threats, i.e. traffic floods, source address spoofing and DoS. The mechanisms adhere to the basic principles of PRGW design and limit all the changes to network edges. This keeps the deployment of PRGW simple, as the upgrade only takes place at the edge nodes, and can be performed one network at a time. We argue that it may possible to take a clean-slate approach, and design a better architecture free of any security weaknesses, at the cost of a huge deployment difficulty. Contrary to this, we take the deployment constraints as the corner stone of our work.

The rest of the paper is structured as follows. Section II discusses the related work. Section III presents vulnerabilities of PRGW in handling the inherent Internet threats. Section IV establishes the basis of our security solutions. Section V and VI describe the security mechanisms and heuristics. Section VII evaluates the security. Section VIII presents the discussion, and Section IX concludes the paper.

II. RELATED WORK

The introduction of NAT at network edges extended the IPv4 address space lifetime. NAT effectively hides the private realm, such that hosts in the private network share a set of public IP address(es) towards the Internet. By default, NAT devices allow outbound connections towards the Internet and create a state to admit subsequent inbound packets of the flow. The connection state enables address translation on packets traversing across the

ISBN 978-3-901882-83-8 © 2016 IFIP

public and private realm, and at minimum contains a 5-tuple: IP and port pair towards public Internet; IP and port pair in the private network; and the transport protocol. Inbound packets that do not have a state in NAT are dropped [8]. As a consequence, connection attempts from the Internet hosts towards the private realm fail, raising the reachability challenge. The current NATs thus employ static port forwarding, or complex NAT traversal mechanisms to admit new connections in the network.

The traditional NAT traversal mechanisms do not scale well to mobile devices [3, 4]. While, static forwarding in NATs can be vulnerable to ills of the Internet, in particular: spoofed flows, network/port scans, and traffic floods from botnets.

Many proposals have attempted to tackle address spoofing and DoS floods. Ingress filtering [9] is a typical solution to the problem of source address spoofing. However, the solution has not been globally adopted, possibly because costs and benefits of ISPs are not well aligned: the receiver or its ISP benefit from spoofing elimination while the other entities bear the expense of configuring and executing the ingress filtering.

IETF proposed the use of SYN Cookies [10] during TCP handshake, to protect the victim host against resource exhaustion from spoofed SYNs. SYN cookie delays the allocation of TCP resources in the host until the sender is verified as *non-spoofed*.

Besides eliminating spoofing, IP puzzles [11] dis-incentivise spurious connection attempts from hosts. The mechanism slows an aggressive host, by requiring the sender to process a received challenge with certain computational effort before it can establish a connection. Similarly, Hop-Count Filtering [12] aims to protect against SYN floods, by comparing the statistics of the received traffic with traffic observed during normal periods. However, these techniques are not in wide use.

Today, an advanced attacker often tricks a large number of hosts to unknowingly participate in launching a DDoS. The *compromised* hosts are mostly bot controlled by the hacker, in a master-slave configuration. Networks typically detect attacks using a set of security approaches, categorized into: Signature detection, Anomaly detection, or a hybrid of both approaches. Upon detecting a DDoS, DoS mitigation proposals typically react by rate limiting the accepted traffic [13]. While it affects the legitimate traffic as well, trace-back techniques are used to locate the malicious entities. An identified attacker is blacklisted and eventually filtered in the admitted traffic.

The research in [14] leverages this understanding of network security to propose a cooperative Feecod architecture. Under this architecture, when a host detects DoS, i.e. from overloading of its resources, the edge router of its ISP rate limits the admitted traffic, so that the total workload for the victim is below its upper bound. A log of each forwarded packet is then sought from the outbound edge to ascertain if no attack originated from its network, upon which the rate-limit is removed. The architecture however requires many changes in end-hosts, as well as in the sender and destination networks, detrimental to its adoption.

The research in [15] tackles DoS through an overlay network that registers the inbound requests before forwarding them to the destination. The proposed indirection infrastructure aims to tackle DoS using P2P networks. The paper in [16] presents various server specific DoS mitigation techniques that require changes in end hosts.

Mobile networks rate availability as the top concern due to high volume of DoS and network/port scans, and typically rate-limit or reset the connections from aggressive hosts [13, 17].

PRGW presents an architecture to overcome the drawbacks of classical NAT traversal solutions. It follows the behaviour of NATs for outgoing connections, such that private hosts connect to Internet sharing a set of public IP addresses. But unlike NATs, it allows Internet hosts to unilaterally initiate connection towards the private hosts using a circular pool of public IP addresses (CPPA). Upon receiving a DNS query for fully qualified domain name (FQDN) of the private host, it temporarily allocates a public IP address from the pool to represent the host in the Internet and creates a temporary *half connection state* that allows forwarding of the subsequent inbound flow to the private host. The client typically initiates the data flow, upon resolving the domain. Upon receiving the first inbound packet from the client, PRGW creates a *full connection state* for the flow and returns the allocated public address to CPPA for future allocations. In this manner, by dynamically assigning an address from CPPA, PRGW protects the private network from direct exposure to the Internet, compared to port forwarding possible in NATs. The half connection state in PRGW applies *endpoint independent filtering* [18] relative to the client, while in the full connection state the filtering is upgraded to *address and port dependent* relative to the client. Since PRGW does not require any changes in end-hosts or remote edge, it avoids the deployment challenge.

III. Security Vulnerabilities

This section analyses the impact of Internet's weaknesses in handling address spoofing and network floods on PRGW. We argue that PRGW does not introduce any explicit security weakness in comparison to the current Internet model, or how NAT allows inbound connections. Like NATs, it also filters to drop the packets that do not have an ongoing connection or a valid state. In addition, 1) the CPPA prevents the private hosts from direct exposure to the Internet, compared to static NATs; and 2) hosts in the private realm are only accessible through their FQDNs, which provides defence-in-depth, in combination with a set of mechanisms that we will introduce in this paper.

Fig. 1 identifies a set of hazardous scenarios where PRGW and the hosts located behind it could be vulnerable to Internet abuses, i.e. in the absence of security mechanisms.

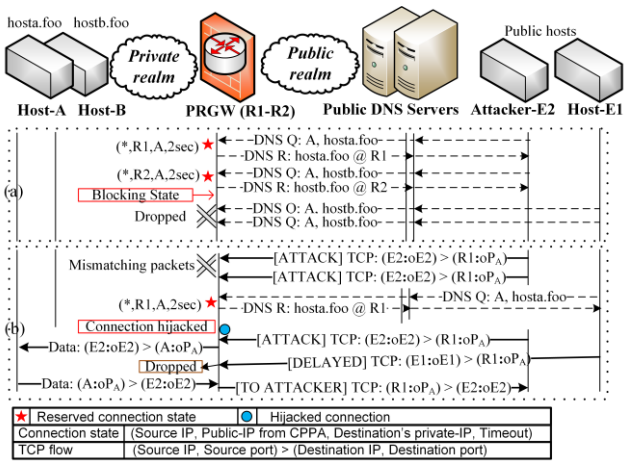


Fig. 1. RGW vulnerabilities to inherent Internet threats

Here we adopt the filtering classification developed in RFC 4787 related to the servers, and use the terms in relation to the clients.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

1) Denial of Service: Fig. 1.a illustrates DoS attack from an *aggressive* Internet host that forces PRGW into *blocking* state, by issuing flood of DNS requests for the served hosts. In this state, the CPPA is depleted due to allocation of all its addresses, leaving PRGW unable to accept new incoming connections. The exhaustion of circular pool could also happen due to poor provisioning of the public IP address pool.

2) Connection hijacking: The attacker in Fig. 1.b floods an address R_1 of PRGW. PRGW would drop any packet unrelated to an ongoing connection or valid inbound state. However, on the event that a public host initiates a connection and address R_1 is allocated, there is a window of opportunity when the attacker can claim the connection state. This will result in DoS to the host that originally requested access to the service. Unfortunately, IP address filtering is not a fail proof solution due to the possibility of source address spoofing in the Internet.

IV. Principles Of Security Mechanisms

Attackers often exploit the best effort nature of the current Internet to launch attacks. Disguising under a spoofed identity, the attackers can successfully inject the traffic in the destination network and yet escape the network auditing. Therefore, the key to improve Internet security comes from deploying mechanisms that eliminate spoofing, authenticate the sender, detect malicious hosts, thwart hijacking attempts and thereby grant access only to the legitimate hosts. We define that PRGW must comply with the following principles to tackle the inherent Internet threats:

1) Flow acceptance must be limited to verifiable sources to tackle address spoofing and prevent resource exhaustion.

2) UDP flow initiations are admitted only after a connection has been signalled through a secure channel e.g. SIP(S) [19].

3) To favor deployment, security algorithms and operations shall not require changes to end-hosts, protocols, or application.

4) Under the network stress, resource access should be granted based on the source reputation.

V. Preventing DNS Abuse/exploitation

PRGW allows unilateral connection initiation to the private realm using CPPA. Since CPPA relies on the inbound domain resolutions, the architecture of PRGW carries a DNS leaf node that is authoritative for the domains located in its private realm.

The state of the art with DNS is such that it uses UDP as transport protocol for majority of its operations. As connection-less protocol, UDP is open to possibility of address spoofing. Attackers often exploit this vulnerability to launch DNS floods, and yet avoid the network audits. Alternatively DNS floods may originate from non-spoofed hosts, under bot control. In addition hackers often use freely-accessible open DNS resolvers, such as Google DNS, as DNS reflectors in launching their attacks.

PRGW is susceptible to this abuse of DNS that can lead to exhaustion of the CPPA resources. To trace *aggressive* host, the current practice in public name servers is not to serve recursive domain requests. As a result, source address of the actual DNS resolver is revealed to the destination. However, the possibility of address spoofing hinders the ability of the destination, i.e. in our case PRGW, to protect itself from DNS floods initiated by the *invisible* attacker.

The resource [20] describes best practices and existing state of the art in the DNS security. Among others, it recommends DNS resolvers/servers to rate limit the domain requests from a source, handle malformed packets, filter requests to not-hosted domains, apply ingress filtering, detect dictionary DNS attacks from hackers i.e. scanning their targets, use of DNSSEC, access control lists (ACL) to filter DNS requests from un-allocated or reserved address spaces, and to drop domain requests originated outside of its network to avoid becoming a DNS reflector.

While these recommendations aim to improve the Internet's resilience against DNS abuses, the ultimate outcome depends on their global adoption by all the network administrators and operators. Realizing this, we attempt PRGW security against the DNS abuses and exploitations by defining set of heuristics and mechanisms, limiting all the changes to the network edges.

A. DNS Relay

We implemented DNS-Relay as a frontend to protect PRGW from direct exposure to the Internet. This is to prevent the CPPA exhaustion from malicious domain resolutions, e.g. inbound DNS floods and spoofed requests. Under this model, PRGW is protected by virtue of *delegating* the DNS security to its ISP.

The DNS relay implementation draws upon the use of DNS reverse proxies in ISP networks and security solutions that aim to secure networks against DNS abuses. In our implementation, we leverage this approach such that the DNS relay forwards an incoming domain request to PRGW and identifies the original sender in the DNS extensions or additional records. The sender tuple identifies: source IP and source port, besides the transport protocol and transaction-ID of the inbound query message. This allows PRGW to identify the original sender, and thus apply its security mechanisms, such as the address allocation model and name server classification. These mechanisms are defined in the subsequent sections. The corresponding DNS response message from PRGW is forwarded by DNS Relay to the actual source, after removing the sender-identification tuple.

The mechanism only requires a few alterations in the edge network, i.e. the ISP name server forwards the inbound domain queries with DNS source information to PRGW. We argue that the changes in the edge network can be motivated by benefits possible from adoption of PRGW, e.g. deployment of servers in private address space, and less-complex session setups. But we consider these aspects beyond the scope of this paper.

The delegation of security to a dedicated DNS-Relay element offers multiple opportunities: 1) it lessens the load of executing the complex DNS security algorithms from PRGW; and 2) the dedicated relay element can independently leverage the existing state-of-the-art and future research in DNS threat detection, to serve the PRGW with legitimate traffic only. As a result, PRGW stays protected against DNS attacks and can allocate the CPPA resources to legitimate hosts.

B. Name Server Classification

When the aforementioned DNS Relay is in attack detection phase, and has not mitigated the DNS attack yet, it is possible that some share of DNS flood is received at PRGW. To prevent the consequent resource depletion, PRGW leverages from the classification of external name servers and allocates the CPPA resources following an Address Allocation Model.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Under this model, PRGW classifies the external DNS servers into: whitelist, greylist and blacklist. Servers on each list are treated differently in PRGW and are promoted/demoted in the classification dynamically, based on the influx of *attack* traffic.

Whitelisting can be based on business contracts and service level agreements (SLAs) between service providers, where the networks that seek *priority* access meet a set of pre-conditions. A whitelist server can meet the specific SLA, by employing the best DNS practices, e.g. active ingress filtering of DNS requests originated in its network, and disabling recursive resolution for external sources. The DNS resolver can also transport domain queries towards PRGW over TCP connection. This eliminates the possibility of spoofing in DNS requests, and on the event that an attack is reported it enables tracing an *aggressive* host back to its network. The terms of whitelisting can be agreed in peering agreements between mobile operators, administrators of the ISPs, or trusted networks, and may stress the networks to employ mechanisms such as DNS/TCP, DNSSEC and ingress filtering to receive whitelist/preferred access.

The whitelist servers are specifically configured in PRGW. By default, the rest of the name servers are *greylist*. This also includes open DNS resolvers and name servers that are freely accessible to Internet hosts, and often serve as DNS reflectors in launching DoS. A greylist name server is therefore offered less resources in PRGW than a corresponding whitelist server.

PRGW actively maintains these lists based on the influx of attack traffic. A name server is demoted to a lower category if states reserved by it repeatedly expire in PRGW. A state expires in the PRGW if it is not claimed by an inbound flow in time T_0. When the state expiration rate for a name server meets threshold R_T, the server undergoes a time penalty T_D in demoted category. A name server that repeatedly exceeds its SLA is blacklisted for time T_B, during which it is barred from accessing the circular pool resources.

C. Circular Pool Address Allocation Model

The CPPA address allocation model responds to an incoming DNS query based on the circular pool load conditions. The model rate limits the number of simultaneous states reserved to a DNS server or for a private host, and manages total allocations of CPPA such that DNS requests from multiple greylist servers only take a portion of the circular pool. For this, the address allocation model operates in conjunction with the name server classification. The model primarily attempts to tackle the DNS floods from *less* secure greylist servers. By prioritizing whitelist servers in address allocation over greylists, the model ensures that whitelist servers always have preferred access to PRGW, particularly under the attack/load conditions.

VI. Filtering Malicious Interent Flows

Internet hackers distribute malicious packets, initiate traffic floods and perform network/port scans to launch their attacks. A hacker can either employ a spoofed identity or hire bots from bot-rental business to launch these attacks. In this section, we introduce a set of mechanisms that attempt to ensure that only a legitimate host gains access to the private realm.

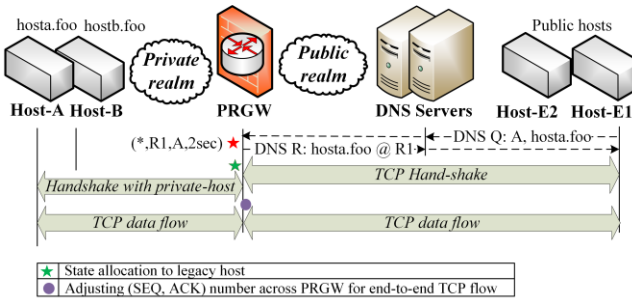


Fig. 2 TCP-Splicing in PRGW

A. TCP-Splice Mechanism

The mechanism ensures that PRGW is secured against hijack attempts from spoofed sources. Fig. 2 presents the mechanism, where an inbound SYN that corresponds to a temporary state is challenged by PRGW with a cookie. Since the TCP handshake only completes on arrival of an ACK bearing the sent cookie, it ensures that the sender is non-spoofed. Next, the PRGW assigns the state to the sender followed by the connection setup with the private host.

Since TCP connection does not complete with spoofed host, PRGW is protected against spoofed sources. PRGW employs a slightly tailored SYN cookie algorithm [10] for computing the initial sequence number (ISN), which is used as a cookie to eliminate address spoofing in the inbound packets.

ISN = time mod-32[5-bits]+MSS encoding[3-bits]+hash{source-IP, destination-IP, source-port, destination-port, SECRET} [24-bits] (1)

The use of the SYN cookie requires that TCP flow is relayed across PRGW. The relay itself must adjust the SEQuence and the ACKnowedgement number on both sides of the PRGW, to maintain the end-to-end semantics of the TCP connection. This is necessary due to the selection of random initial sequence numbers by the private host and PRGW. The translation of SEQ and ACK numbers effectively splices the connection on both sides of the PRGW. By keeping the SEQ number of the SYN to the private host the same as that of the inbound SYN, PRGW saves translation cost on one TCP sequencing.

B. Bot-detection Scheme

Attacks to PRGW could also originate from non-spoofed, i.e. bot hosts. In this section, we present a bot-detection method that attempts to protect PRGW against SYN floods from botnets, and thus complements the limitations of TCP-Splice.

In contrast to the networking elements that simply filter the packets mismatching to a flow or a connection state, PRGW can carry *bot-detection on the dropped packets*. Fig. 3 illustrates the mechanism where PRGW seeks to ascertain if the sender of the repeatedly mismatching SYNs is a non-spoofed entity. When the mismatching packets exceed a threshold in time T_O, PRGW handles the next inbound SYN failing to claim a state as per the SYN cookie algorithm. The subsequent arrival of an ACK bearing the sent cookie establishes the sender as *non-spoofed*. The history of dropped packets together with the non-spoofing check hints at a high likelihood of the sender as a *bot-controlled host*. Following which, the PRGW refuses any state to this host.

1) Implementation Considerations

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

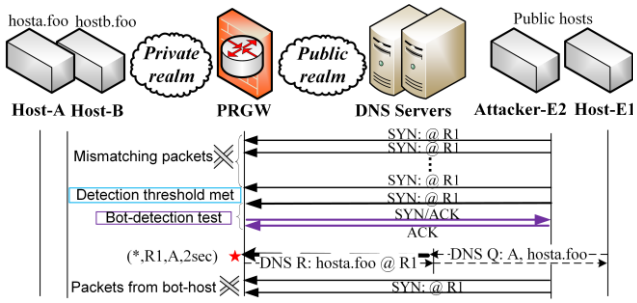


Fig. 3 Bot-detection method on SYN floods from a bot-operated host

An attacker meets the detection threshold, when mismatched packets reach a threshold in time T_O. Attackers typically initiate SYN floods at higher rates than a normal host, which only re-attempts if the previous packet is not responded within a retransmission timeout (RTO). RTO is typically an operating system defined parameter, and we choose a value below it as measurement interval T_O, since it significantly differentiates the legitimate behaviour from an attack. For TCP, UNIX domain sockets and Windows define RTO as 3 seconds [21]. Hackers can also initiate slow-rate SYN floods from various addresses, and thus bypass the bot-detection threshold. This will reveal the lower bound of PRGW security, where PRGW is secure against spoofed flows only. Bot-detection is executed only after an attacker meets the detection threshold, because a continuous monitoring for bot-detection would be too costly.

To realize the impact of our design choices, we classify the source of a mismatching packet into: 1) spoofed host; 2) non-spoofed attacker; or 3) a legitimate host. A packet may arrive from a host whose corresponding state was previously hijacked. However, a legitimate host does not re-attempt (or would not re-attempt x times) within RTO, and thus it would not meet the detection threshold. Similarly, a spoofed address cannot reply to SYN/ACK with the sent cookie, and hence is not blacklisted as attacker. Thus, only a bot-operated host is susceptible to this mechanism after it replies with an ACK bearing the sent cookie.

2) *Caveats and Considerations*

We realize that Bot-detection is not a fail proof solution and is vulnerable to abuse. Thus, we suggest to dynamically adjust the detection threshold and measurement interval T_O, to prevent the exploitation of the protection mechanism. Despite all the countermeasures, the possibility of a false alarm exists, and thus a bot-suspected host is blacklisted for temporal time T_D.

Since both the TCP-splice and Bot-detection could co-exist in the PRGW, there is a need to differentiate an inbound ACK under Bot-detection from an ACK that is part of TCP handshake with a public host. For this, SYN cookies of TCP-Splice and Bot-detection must differ, e.g. in *SECRET* value of equation 1.

C. *Security by Deployment*

A carrier-grade realm gateway (CGRG) can improve security of the private realm from a variety of resources at its disposal. For instance, the traffic from white and greylist sources can be accepted over separate sets of interfaces. This is often possible e.g. in mobile networks, where the traffic from other operators or corporate networks is processed on separate interfaces than those for public Internet [13]. This ensures dedicated access for whitelist networks and enables pursuing rather aggressive security on the greylist interfaces.

D. *Enhancing the Circular Pool Algorithm*

In [22], we present a new algorithm for allocating the public IP addresses of the circular pool, enabling fine-grained access control to flows arriving from the Internet. The new algorithm significantly improves the scalability and security of PRGW.

The underlying idea is to address the services and endpoints simultaneously. To that end, we leveraged the concept of the SRV DNS records and created Service FQDN (SFQDN) to address services on end-hosts. Currently, the use of SRV is only limited to a few applications, whereas the DNS A records are widely in use. SFQDN bridges this gap between DNS A records and the SRV records, and defines simple domain names linked to a specific service. For example, an SSH service at Host A – `a.foo` can be represented as `ssh.a.foo` or it can arbitrarily be defined as a combination of port number and transport protocol as in `tcp22.a.foo`. For aesthetic/security purposes, hosts can hide their SFQDN naming in favour of a more user friendly name, e.g. using CNAME records in DNS as a pointer to other domain names. The SFQDN and its mapping to a port can stay inside the PRGW while the CNAME to PRGW mapping is stored at a DNS server in the ISP network.

Since the SFQDN includes both the endpoint and the service, using the RFC defined terminology, SFQDN resolution allows *endpoint independent but port dependent filtering* in the half connection state relative to the remote host. The more specific half state allows reusing a public IP address for several different services, improving the scalability of CPPA. Theoretically, it implies that a single IP address can be reused as many times as the combination of available ports and protocols. Meanwhile, forcing the blocking state on PRGW becomes more difficult because the hacker must send significantly larger number of DNS requests to reserve the address pool for all the ports. In addition, the hackers must also target the allocated port besides simply flooding the public IP addresses for state hijacking. The temporary half connection state (R_X:oP_H, H:iP_H, P_{proto}, $T_{timeout}$) is unique and carries the IP address and port of the private host (H:iP_H), IP address and port on the public side of the PRGW (R_X:oP_H), the protocol (P_{proto}) and lifetime ($T_{timeout}$) of the entry. Upon the arrival of the first packet of the flow, PRGW upgrades the filtering to address and port dependent.

SFQDN contributes to security due to its more specific address allocation. This increases the attack surface, such that a hacker has more opportunities to meet the detection threshold, as a hacker must scan the entire port range to discover the active services and compromise respective allocations. The increased scalability also makes it more difficult to force the blocking state. Since PRGW solely admits inbound connections based on the domain queries, it becomes simple to temporarily block a service under attack and collect the evidence of misbehaviour.

VII. SECURITY EVALUATION

This section evaluates the security of PRGW in tackling the inherent Internet threats: source address spoofing, network/port scans and DNS floods. We implemented the above mechanisms in our PRGW prototype and subjected them to a set of *attacks* to determine the bounds of the PRGW security. The prototype runs in our test network, which is built in Linux environment using standard Linux networking capabilities: linux containers

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

and switches. The PRGW node in the testbed attends hosts and services located in its private realm, whereas legacy hosts in the testbed either initiate inbound connections or attacks towards the PRGW. The legacy nodes use virtual network interfaces to provide an illustration of many hosts participating in the traffic towards the PRGW.

We utilize Scapy [23] to craft *malicious* packets and launch attacks on PRGW. For our testing, this enables the legacy nodes to: 1) initiate spoofed traffic; and 2) emulate network floods from non-spoofed hosts, i.e. bots. The attack load is measured in SYNs per second from the hacker, whereas the network delay between the nodes is artificially generated. The outcome of the testing reveals the effectiveness and cost of the PRGW security, in terms of the ratio of the hijacked connections and processing delay introduced in the PRGW, respectively.

Fig. 4 demonstrates the PRGW security against DNS abuses. Having pre-configured the whitelist servers, we submit PRGW to DNS flood from multiple greylist servers. In the absence of security, the DNS flood would reserve all the CPPA resources and thus force PRGW in blocking state. However, the address allocation model notes that the DNS source is greylist and limits the resource allocations to a portion of the circular pool.

In this manner, the allocation model prevents the exhaustion of CPPA under DNS floods and ensures that whitelist servers have access to PRGW even under load conditions. A similar resource depletion attack using SFQDN is more challenging, since the high flood rate and amount of domain queries required to force blocking state increases likelihood of attack detection. Moreover, the rate limits on simultaneous domain queries from a DNS server and to a host, hinders the attacker ability to launch DNS floods from a few name servers or open resolvers.

We tested the CPPA enhancement algorithm by designing different inbound traffic patterns that evaluate the improvement in PRGW security due to SFQDN, especially against network and port scan attacks. We designed the following tests:

- Test1: 100% of the inbound traffic has the FQDNs of the destination hosts. On the event that hacker's packet meets an allocated address, the half connection state is claimed.
- Test2: 50% of the inbound traffic is generated using FQDN and the rest employs SFQDN. Hacker must target the right IP and port pair to claim the SFQDN allocation.
- Test3: 75% of inbound traffic is SFQDN; the rest FQDN.
- Test4: 100% of the inbound traffic is SFQDN.

Fig. 5 shows the result of stressing the prototype with above traffic patterns at network delay of 200 msec and a constant load of 4 connections per second. The connection load is distributed among private hosts and follows an exponential distribution. In parallel, a network scan attack at 40 SYNs/sec from the legacy nodes targets the CPPA. The figure reveals that for test1: FQDN initiations only, nearly all the connections are hijacked. This is because the hacker constantly scans the CPPA at high rate and beats the legitimate host in claiming the end point independent state. However, as the share of SFQDN grows and nears 100% in total inbound DNS queries, the ratio of hijacked connections declines and nears zero for an all SFQDN traffic. This is due to the fact that besides scanning the public IP addresses, a probing attacker also has to randomly scan for the allocated port out of 2^{16} possible ports to claim the state. The more specific address allocation for SFQDN enables more opportunities for a hacker to meet the detection threshold, which leads it to blacklisting in Bot-detection and subsequent rise in the legitimate connections.

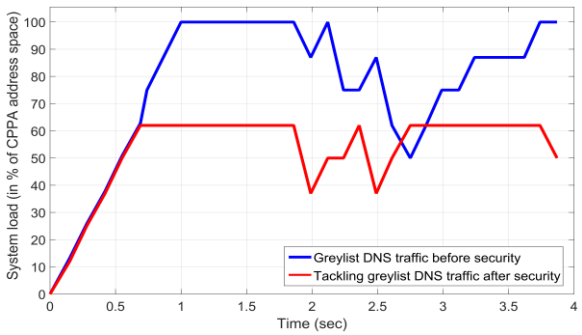


Fig. 4 Allocation model limiting the DNS flood from greylist servers

Next, we evaluate the PRGW security against spoofed flows and network scans. We subjected PRGW to 3 connections (i.e. DNS requests) per second and in parallel launched 40 spoofed SYNs per second from the legacy nodes to CPPA, for hijacking the states. The testing reveals that spoofed SYNs failed to claim the half states due to better filtering enabled by the SFQDN. However, the spoofed SYNs could hijack the FQDN allocations in the absence of security mechanisms, because a hacker would scan the network at a high rate and can compromise states if its packet meets an IP address, allocated in the FQDN state.

In contrast, TCP-Splice successfully thwarts hijack attempts from spoofed sources and prevents leaking of spoofed packets into the private realm. Fig. 6 summarizes the PRGW's delay in assigning the half state to legacy hosts, not considering the link latency. The figure shows that TCP-Splice obviates spoofing in the admitted flows, at the cost of delaying the claim to the half

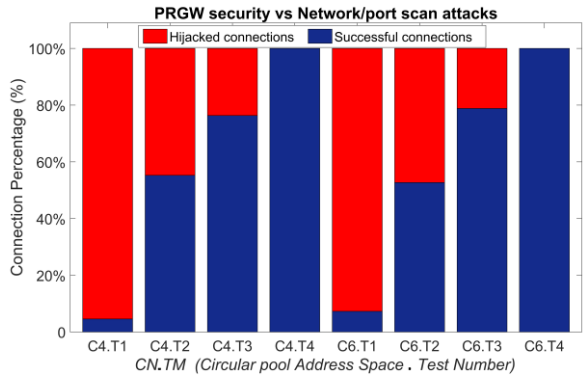


Fig. 5. Impact of inbound traffic type on security, versus network scans

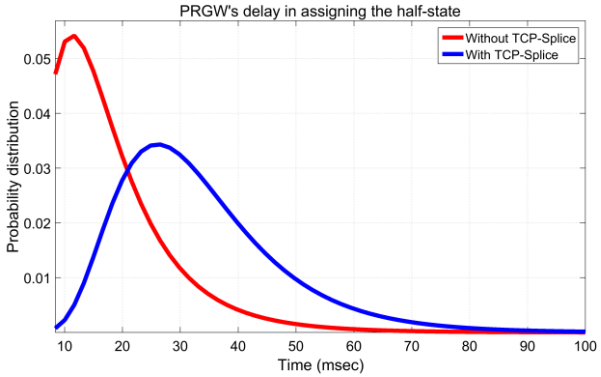


Fig. 6 Delay in assigning TCP half-connection state, before and after security

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

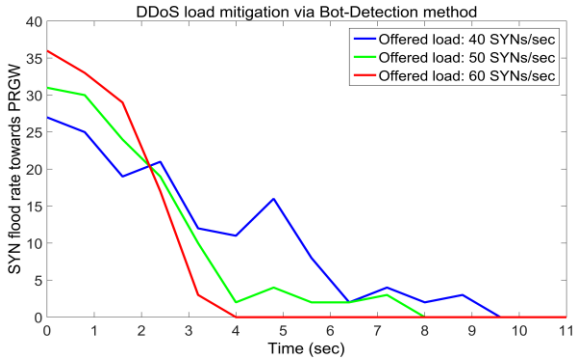


Fig. 7 Mitigating DDoS (SYN flood) via Bot-Detection method

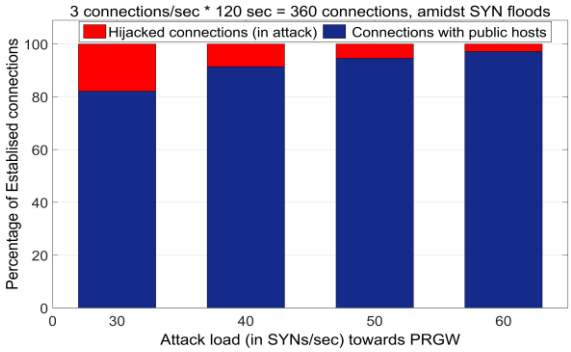


Fig. 8 Securing states against SYN floods from bot-controlled hosts

connection state. This is because to its SYN the sender receives a cookie from PRGW, which must be relayed back in the next inbound ACK to establish the connection, causing the delay.

In terms of performance, this limits the reusability of the public IP address and the port combination by the same duration for the next inbound connection. In a real network, the end-to-end latency for TCP messages would be added to compute the total delay in assigning the half-state. It is possible to reduce the average delay penalty caused by TCP Splice by using it selectively, i.e. on privileged ports, or under network attacks.

Fig. 7 presents an overview of PRGW security against SYN floods from bot hosts, which are non-spoofed sources under a botnet. Without security, an attacker can constantly scan the CPPA at high rate and on the event that its packet meets a half connection state, it will claim the allocation. In comparison, the Bot-detection would constantly track the dropped packets and once they exceed a threshold, the source is blacklisted following a non-spoofing test. As a result, states reserved by legacy clients are protected against the hijacking attempts. The figure shows that Bot-detection is more reactive to high flood rates and filters them earlier, as they quickly meet the detection threshold.

Fig 8 expands on the same result and shows the impact of stressing PRGW with a SYN flood sourced from eight hosts participating in the attack. In parallel, the public hosts initiate 3 connections/second towards the CPPA of three addresses, the network delay is 200 msec and the bot-detection threshold for a source is 12 dropped SYNs in 2 second interval. In practice, this threshold could be chosen during network planning phase, i.e. based on peaks in the traffic statistics graph. The figure reveals that the ratio of hijacked connections decreases as the attack load increases, since an attack with more active bots is filtered earlier, contributing to rise in the legitimate connections. Fig. 9 shows the impact of network delay, where the network delay is

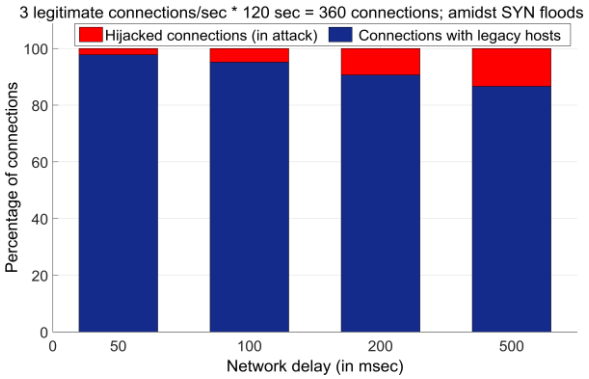


Fig. 9. Impact of network delay on PRGW security

time elapsed from creating a half connection state to the arrival of first packet from the client host.

The outcome of Bot-detection depends on multiple factors. From attack perspective, these are: number of flooding sources; choice of network/port scan strategies, i.e. targeting the known services or random port scans; and flooding rates for attacks or avoiding the detection threshold. On the other hand, the PRGW can improve its defense by dynamically adjusting the detection threshold, allocating more circular pool addresses and allowing SFQDN only. These strategies can provide more opportunities to hackers to meet the detection threshold and get blacklisted.

The paper obviously cannot present the PRGW security as a function of all the parameters. But, the testing generally reveals that Bot-detection reacts the best when attack volume is shared by few hosts. This means that to succeed a hacker must sacrifice rather large number of bots that do not use spoofing, and hence are likely to be identified by the target network's PRGW. The use of Bot-detection together with TCP-Splice guarantees that only legitimate hosts gain access to the private realm.

Fig. 10 compares the security of FQDN initiated connections in PRGW, in presence and absence of the security algorithms. Again,-we subject the PRGW to a load of 3 connections/sec at a network delay of 200 msec, while 8 non-spoofed sources flood CPPA with 40 SYNs/sec. Fig. 10.b shows that the ratio of hijacked connections decreases significantly after the security. The figure also reveals the impact of increasing CPPA address space, which contributes to security by increasing the overall attack surface. This shows that careful network planning and proper dimensioning of the CPPA resources can have positive impact on the PRGW security.

To deeply analyse the security of SFQDN states, we divide the Internet hackers into: 1) probing/scanning hackers; and 2) advanced hackers. A probing hacker scans the entire CPPA address space and port range to discover the available services, IP addresses or NAT mappings. It is quite likely that such an attacker due to its limited victim's knowledge, and thus random network scanning will fail to attack PRGW as shown for Test 4 in Fig. 5. In comparison, an advanced hacker may already know services/ports in the target network, via knowledge sharing among hackers or using botnets that perform the service discovery process. As a result, the hacker can target the SYN floods to the specific ports. We analysed the security of SFQDN allocations against such attacks and depict it in Fig. 11. We use the same test parameters as for Fig. 10. The result in Fig. 11.b shows a rise in the legitimate connections after security. This is because Bot-detection filters the hosts that initiate the floods

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

towards PRGW, however it is possible that a flood hijacks some states before it is entirely mitigated, as shown in the figure.

Clearly PRGW attains best-case security, when the hacker is unaware and simply scans the network for vulnerable services or IPs, i.e. a probing attacker, while the PRGW accepts SFQDN requests only. Under the premise that the attacks are directed to the served ports, it is perhaps best that SFQDN naming is changed to new service ports. This will force attacker to restart its service/port discovery cycle and help PRGW regain its best case security. Such use of SFQDN is possible in cases where a single administration owns or manages both the remote hosts and the PRGW. For example, Internet of Things (IoT) can emerge as one such use case where the communicating nodes and gateway will fall under single administration. In absence of such a scheme, Fig.11 shows the security of SFQDN allocations against an advanced hacker.

It is pertinent to mention that in our testing no state allocation was compromised by spoofed flows. However, few allocations were hijacked by the packets from the bot-hosts. This is because before a traffic flood is mitigated, some of its packets can beat a legitimate host in claiming the allocated state, and cause DoS to the actual client. Thus the security of PRGW can exhibit false negatives during attack. However, these false negatives reduce as the attack progresses, since the more active bots will be filtered upon exceeding the detection threshold.

The ratio of false negatives can further reduce by: 1) network dimensioning that presents an attacker more opportunities to meet the detection threshold; and 2) dynamically adjusting the detection threshold to prevent exploitation of the protection mechanisms. Though our testing identified few false negatives, PRGW did not exhibit any false positives, i.e. classifying a valid client as attacker. We argue that in the PRGW networks, a false negative is not as severe as a false positive; since a client that suffers hijacks can always re-attempt to access the desired service in the private realm.

Table-I summarizes the mechanisms deployed for securing PRGW against Internet threats and their impact on the PRGW's performance. Whereas, Table-II presents the duration that a received packet is processed in the PRGW security before a decision is reached. The delay values in Table II are computed within PRGW at algorithmic level, i.e. they do not include the time spent in acquisition, packetizing and forwarding of the packet. These values nicely fit with the delay requirements of the end-to-end connection. Hence, PRGW and its hosts can be protected at the cost of minute processing delay.

TABLE I. SECURITY MECHANISMS AND THEIR PERFORMANCE

Security threats	*Mechanisms*	*Cost of Security*
Source address spoofing	TCP-Splice	Extends duration of assigning the state
Bot-controlled flows	Bot-detection	Possible False Negatives
Malformed ACK segments	cookie verification	-
DNS-floods	Rate limit simultaneous DNS allocations to hosts and greylist server(s)	Less trusted servers face congestion, under load
Spoofed DNS requests	DNS/TCP, DNS Relay and Ingress filtering	SLA negotiations, and sender's effort

TABLE II. PROCESSING OF INBOUND PACKET/FLOW IN THE PRGW SECURITY

	Processing delay	*Outcome*
Inbound TCP SYN segment	< 0.1 msec	Respond with cookie
TCP-Splice (on non-spoofed)	~1 msec	Eliminates spoofing
Packet not matching any state	~0.01 msec	Processing in bot-detection method
Malformed ACK segments	< 0.1 msec	Accept/Drop
DNS/TCP request	Connection-setup delay for 1st query	Spoofing elimination in the DNS queries
(Greylisted) DNS/UDP request	~ 1 msec	Accept if the load < threshold

The current PRGW prototype employs a minimalistic set of rules, i.e. rate-limiting, to provide the firewall functions. The deployment of PRGW at the network edges would require integrating PRGW with a commercial firewall. We argue that integrating a firewall would further reduce and nearly eliminate the false negatives during an attack, besides hardening the security of PRGW against well-known attacks.

VIII. DISCUSSION

The security testing shows promising results. Though, the implemented mechanisms exhibit false negatives, the proposed firewall integration will present PRGW as a feasible network function. For HTTP, which can set up many flows after a single DNS query, PRGW employs an HTTP reverse proxy to serve the inbound requests. Besides lessening the load on the circular pool, it offers advantages in terms of offloading SSL encryption and load balancing to the proxy [4]. Compared to proxy-server operations in SOCKS [24], TCP-Splice offers an efficient redirection mechanism for admitting the flows, and moves the processing load from caching at application-layer to mere sequence number translation at the transport layer.

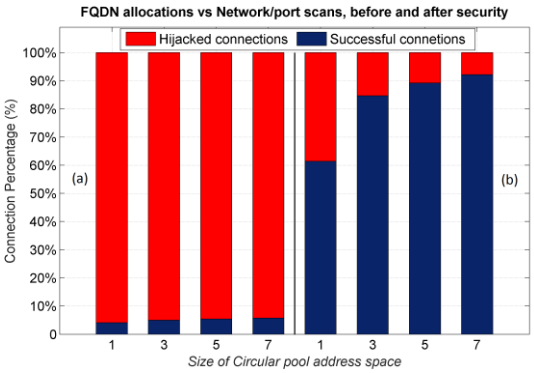


Fig. 10. Security of FQDN allocations, *(a)* without and *(b)* with security

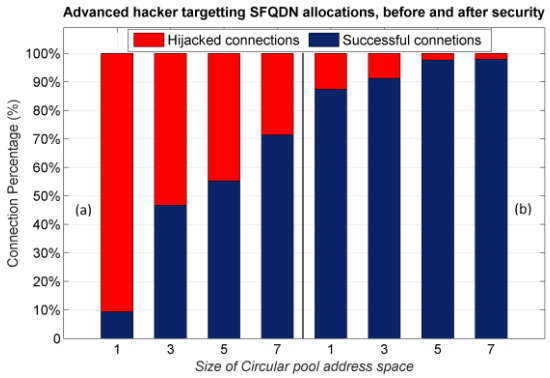


Fig. 11. Security of SFQDN allocations against advanced hackers, *(a)* without and *(b)* with PRGW security

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

In [4], we introduced PRGW to address the challenges in the Internet and offer a reachability solution that overcomes the drawbacks of the classical NAT traversals. The contribution of this paper is in presenting PRGW as a feasible function in the edge nodes that is well protected against the Internet attacks.

For end host security we can compare PRGW to the case that the application is using the cumbersome but functional IETF NAT traversal mechanisms [2]. To prevent attacks to the hosts that use SFQDNs and to identify the host application, we see the need to integrate an application policy database in PRGW that will link SFQDNs to application parameters, such as proxy name or addresses that can communicate with this SFQDN, and timeouts that will be used to monitor the application traffic, etc. PRGW can consult this database for making address allocation decisions. The idea would be to allow flows only from known entities or allocate most CPPA resources to known entities. The time parameter in the database can also rate limit an application that assumes connection initiation from unknown entities. We believe this would work for example for Peer-to-Peer SIP.

By tying the use of communication service proxies to PRGW via an application policy database, and by monitoring and rate limiting the application traffic, we reach the same level of host protection as in the case of application-specific NAT traversal.

IX. CONCLUSION

PRGW offers better than NAT service to hosts in the private address space. Unlike NAT, it presents a scalable way to initiate flows from other networks to hosts in the private address space. At the same time, no application-layer NAT traversal code is needed. Private hosts can stay reachable without need for keep-alive signalling to maintain their state, thus reducing the battery consumption. It offers shorter session setup delays, and eases configuring and managing of the port forwarding compared to how it is implemented in NATs, since PRGW can dynamically establish it upon the domain resolution.

This paper complements these advantages of PRGW through a security analysis that presents it as a feasible Internet function. The presented heuristics and mechanisms harden the PRGW against the inherent Internet weaknesses, such as source address spoofing, network/port scans and DNS floods. The mechanisms limit all the changes to network edges to favour the deployment and prevent the resource exhaustion in PRGW, by limiting flow acceptance to verifiable sources only.

PRGW admits inbound connections towards private hosts based on the domain name resolutions. We briefly discuss the current state of the art with DNS and leverage it for securing PRGW against Internet DNS abuses. Besides employing the best practices, we also present a new Bot-Detection algorithm that together with TCP-Splice attempts PRGW security against flows from spoofed and non-spoofed sources.

The security evaluation reveals that PRGW can be protected against the inherent Internet threats, at the cost of minimal processing delay. We briefly discuss the impact of different factors, such as attack strategy and inbound traffic pattern on the effectiveness of PRGW security. By addressing the security limitations of PRGW, this paper further adds to the claim of deploying PRGW at the network edges to address the Internet challenges [4]. We argue this further by briefly comparing NAT and PRGW, and the security of end hosts under both solutions. The adoption of PRGW to networks is simple, since it does not require any changes in end hosts, protocols or applications.

REFERENCES

[1] ITU-T ICT STATISTICS. Free statistics. [Retrieved on Oct.2015] Available: https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf

[2] L. Daigle, IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation, RFC 3424, Nov 2002

[3] G. Camarillo, J. Mäenpää, A. Keränen and V. Andersson, "Reducing Delays Related to NAT Traversal in P2PSIP Session Establishments," in Proc. IEEE Consumer Communications and Networking Conference, CCNC 2011, pp.549-553, Las Vegas, NV, USA, 9-12 Jan. 2011.

[4] J. Llorente, R. Kantola, N. Beijar, and P. Leppäaho, "Implementing NAT Traversal with Private Realm Gateway", Communications (ICC), 2013 IEEE International Conference, 2013, pp. 3581-3586.

[5] R. Kantola, "Implementing Trust-to-Trust with Customer Edge Switching,", AMCA in connection with AINA 2010, Perth, Australia, 20-23 April 2010.

[6] H. Kabir, R. Kantola, and J. Llorente, "Security Mechanisms for a Cooperative Firewall," in Internatinal Symposium on Cyberspace Safety and Security (CSS), Paris, 2014.

[7] "2014 Cisco Annual Security Report," CISCO, 2014.

[8] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.

[9] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827, May 2000.

[10] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987, August 2007.

[11] M. Ma, "Mitigating Denial of Service Attaks with Password Puzzles," in Information Technology: Coding and Computing, 2005, pp. 621-626.

[12] H. Wang, C. Jin, and K. G. Shin, "Defense Against Spoofed IP Traffic using Hop-Count Filtering," in IEEE/ACM, Transactions on Networking, Volume 15, 2007, pp. 40-53.

[13] "SRX Series AS Gi/SGi Firewall for Mobile Network Infrastructure Protection," Juniper Networks, Whitepaper.

[14] H. Beitollahi and G. Deconinck, "A Cooperative Mechanism to Defense Against Distributed Denial of Service Attacks," in 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) 2011, 2011.

[15] R. Lua and K. C. Yow, "Mitigating DDoS Attacks with Transparent and Intelligent Fast-Flux Swarm Network," in IEEE Networks, Volume: 25, Issue:4, 2011, pp. 28-33.

[16] R. R. Robinson and C. Thomas, "Evaluation of Mitigation Methods for Distributed Denial of Service Attacks," in 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2012, pp 713-718.

[17] "DEFEATING DDOS ATTACKS," CISCO Systems, Inc., White Paper, 2014.

[18] F. A. Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," RFC 4787, 2007

[19] J. Rosenberg, et al., "SIP: Session Initiation Protocol," RFC 3261, 2002

[20] "DNS Best Practices, Network Protections, and Attack Identification," CISCO Systems, White Paper, 2015.

[21] MICROSOFT. TCP/IP and NBT configuration parameters for Windows. [Online]. http://support.microsoft.com/kb/314053 {On: 22.07. 14}

[22] J. Llorente and R. Kantola, "Transition to IPv6 with Realm Gateway 64," IEEE International Conference on Communications (ICC), London, June, 2015.

[23] (2015, Mar.) SCAPY. http://www.secdev.org/projects/scapy/

[24] M. Leech, M.Ganis, Y. Lee, R. Kuris, D Koblas, L. Jones , "SOCKS Protocol Version 5," RFC 1928, March 1996.

Verified iptables Firewall Analysis

Cornelius Diekmann, Julius Michaelis, Maximilian Haslbeck, and Georg Carle
Technische Universität München
Email: {diekmann|carle}@net.in.tum.de, {michaeli|haslbecm}@in.tum.de

***Abstract*—We present a fully verified firewall ruleset analysis framework. Ultimately, it computes minimal service matrices, i.e. graphs which partition the complete IPv4 address space and visualize the allowed accesses between partitions for a fixed service. Internally, we are working with a simplified firewall model and a core contribution is the translation of complex real-world iptables firewall rules into this model. The presented algorithms and translation are formally proven correct with the Isabelle theorem prover. A real-world evaluation demonstrates the applicability of our tool. Both the `iptables-save` datasets and the Isabelle formalization are publicly available.**

I. Introduction

Firewall rulesets are inherently difficult to manage. It is a well-studied but unsolved problem that many rulesets show several configuration errors [1]–[3]. Tools were designed to help uncover configuration errors and verify a ruleset. We focus on tools for the static analysis of rulesets. They have the benefit that the analysis can be carried out offline, without any negative effects on the network. In contrast to testing, static analysis can achieve a full coverage (e.g. the results hold for all packets) and thus are able to uncover all errors and give strong guarantees for the absence of certain classes of errors. However, in practice, static ruleset analysis tools fail for various reasons: They do not support the vast amount of firewall features, they require the administrator to learn a complex query language which might be more complex than the firewall language itself, the analysis algorithms do not scale to large firewalls, and the output of the verification tools itself cannot be trusted.

To overcome these issues and to foster static analysis and verification of real-world firewall rulesets, we present the first fully verified and large-scale tested Linux/netfilter iptables firewall analysis and verification tool. In detail, our contributions are:

- A simple firewall model, designed for mathematical beauty and ease of static analysis (Section III)
- A series of translation steps to translate real-world firewall rulesets into this simple model (Section IV)
- Static and automatic firewall analysis methods, based on the simple model, featuring
  - IP address space partitioning (Section V)
  - Minimal service matrices (Section VI)
- Full formal and machine-verifiable proof of correctness (Section Availability)
- Evaluation on large real-world data set (Section VII)

The Linux iptables firewall is wide-spread, has evolved over a long time, and is well-known for its vast amount of features. In addition, in production networks, huge, complex, and legacy firewall rulesets have evolved over time. Therefore, iptables poses a particular challenge. Naturally, our methodology can also be applied to firewalls with simpler semantics, or younger technology with yet fewer features, e.g. Cisco IOS Access Lists or OpenFlow.

ISBN 978-3-901882-83-8 © 2016 IFIP

We outline related work in Section II. The real-world and simplified firewall models are presented in Section III. We detail on the translation between these models in Section IV. Afterwards, we present the IP address space partitioning (Section V) and service matrices (Section VI). In Section VII, we evaluate our algorithms on a large set of real-world iptables rulesets.

II. Related Work

We will call the features a firewall can use to match on packets *primitives*. For example, among others, iptables supports the following primitives: src IP address, layer 4 port, inbound interface, conntrack state, entries and limits in the `recent` list, ...

Popular tools for static firewall analysis include FIREMAN [4], Capretta et al. [5], and the Firewall Policy Advisor [6]. They support the following primitives: IP addresses, ports, and protocol. This corresponds to (a subset of) our simple firewall model, hence, these tools would not be applicable to most firewalls from our evaluation. The tools focus on detecting conflicts between rules and can consequently not offer service matrices. The work most similar to our IP address space partitioning is ITVal [7]: It supports a large set of iptables features and can compute an IP address space partition [8]. Unfortunately, ITVal is not formally verified and its implementation has several errors. For example, ITVal produces spurious results if the number of significant bits in IP addresses in CIDR notation [9] is not a multiple of 8. It does not consider logical negations which may occur when `RETURN`ing prematurely from user-defined chains, which leads to wrong interpretation of complement sets. It does not support abstracting over unknown primitives but simply ignores them, which also leads to spurious results. For rulesets with more than 1000 rules, ITVal requires tens of gigabytes of RAM. Finally, ITVal neither proves the soundness nor the minimality of its IP address range partitioning. Nevertheless, ITVal demonstrates the need for and the use of IP address range partitioning and has demonstrated that its implementation works well on rulesets which do not trigger the aforementioned errors. Building on the ideas of ITVal (but with a different algorithm), we overcome all presented issues.

Exodus [10] translates existing device configurations to a simpler model, similar to our translation step. It translates router configurations to a high-level SDN controller program, which is implemented on top of OpenFlow. Exodus supports

many Cisco IOS features. The translation problem solved by Exodus is comparable to this paper's problem of translating to a simple firewall model: OpenFlow 1.0 only supports a limited set of features (comparable to our simple firewall) whereas IOS supports a wide range of features (comparable to iptables); A complex language is ultimately translated to a simple language, which is the 'hard' direction.

Complementary to our verification tool, and well-suited for debugging, is Margrave [11]. It can be used to query firewalls and to troubleshoot configurations or to show the impact of ruleset edits. Margrave can find scenarios, i.e. it can show concrete packets which violate a security policy. Our framework does not show such information. Margrave's query language (which should be learned by a potential user) is based on first-order logic.

III. Firewall Semantics

All facts presented in this work are formally verified with the Isabelle theorem prover [12]. All executable algorithms are also implemented in Isabelle and formally proven correct.

Isabelle is an LCF-style theorem prover: A proposition is only accepted by Isabelle if it can be explained to its mathematical inference kernel. That kernel is very small and well-understood by the formal methods community which makes it very unlikely that Isabelle allows proving false statements. The last 20 years of Isabelle in practice underline this statement. In general, the formal methods community treats facts machine-verified with Isabelle as well-founded truth. Also, the real-world firewall reference model we will use in this work (Section III-B) has been previously evaluated by said community [3]. Our formalization, implementation, and proofs are publicly available (cf. Section Availability). An interested reader can replay the proofs and results of the evaluation on her system. For brevity, in this paper, we omit all technical proof details and only outline the intuition of the correctness proofs. For further mathematical details, we refer the interested reader to our proof document. We use Isabelle's standard Higher-Order Logic (HOL). This means, all proofs can be reduced to the axioms of HOL. We stick closely to the formalization and do not sweep any assumption under the carpet.

Our notation is close to Isabelle, Standard ML, or Haskell: Function application is written without parentheses, e.g. $\mathsf{f}\ a$ denotes function f applied to parameter a. For lists, we denote cons and append by '$\mathrel{::}$' and '$@@$', e.g. '$x \mathrel{::} [y, z] @@ [a]$'. Linux shell commands are set in `typewriter` font. Executable functions are set in sans serif font. We will write firewall rules as tuple $(m,\ a)$, where m is a match expression and a is the action the firewall performs if m matches for a packet. The firewall has two possibilities for the filtering decision: it may accept (✓) the packet or deny (✗) the packet. There is also an intermediate state (?) in which the firewall did not come to a filtering decision. Note that iptables firewalls always have a default policy and the ? case cannot occur as final decision.

A. Simple Firewall

First, we present a very simple firewall model. This model was designed to feature nice mathematical properties but it is too simplistic to mirror the real world. Therefore, we will afterwards present a model for real-world firewalls. Section IV will show how rulesets can be translated between these two models. This preprocessing step simplifies all future static firewall analysis. The model is a simple recursive function. The first parameter is the ruleset the firewall iterates over, the second parameter is the packet.

$$\begin{aligned}
&\mathsf{simple\text{-}fw}\ \ [\,] && p = && ?\\
&\mathsf{simple\text{-}fw}\ \ ((m,\ \texttt{Accept}) \mathrel{::} rs)\ p = \\
&\qquad \text{if } \mathsf{match}\ m\ p \text{ then } ✓ \text{ else } \mathsf{simple\text{-}fw}\ rs\ p\\
&\mathsf{simple\text{-}fw}\ \ ((m,\ \texttt{Drop}) \mathrel{::} rs) && p = \\
&\qquad \text{if } \mathsf{match}\ m\ p \text{ then } ✗ \text{ else } \mathsf{simple\text{-}fw}\ rs\ p
\end{aligned}$$

A function match tests whether a packet p matches the match condition m. The match condition is an 7-tuple, consisting of the following primitives:

$$(\text{in},\ \text{out},\ \text{src},\ \text{dst},\ \text{protocol},\ \text{src ports},\ \text{dst ports})$$

In contrast to iptables, negating matches is not supported. In detail, the following is supported:

- in/out interface, including support for the '+' wildcard
- src/dst IP address range in CIDR notation, e.g. 192.168.0.0/24
- protocol (Any, tcp, udp, icmp, or any numeric protocol identifier)
- src/dst interval of ports, e.g. 0:65535

For example, we obtain an empty match (a match that does not apply to any packet) *iff* an end port is greater than the start port. The match which matches any packet is constructed by setting the interfaces to "+", the ips to 0.0.0.0/0, the ports to 0:65535 and the protocol to Any. With this type of match expression, it is possible to implement a function conj which takes two match expressions m_1 and m_2 and returns exactly one match expression being the conjunction of both.

Theorem 1 (Conjunction of two simple match expressions)**.**

$$\mathsf{match}\ m_1\ p\ \wedge \mathsf{match}\ m_2\ p\ \longleftrightarrow \mathsf{match}\ (\mathsf{conj}\ m_1\ m_2)\ p$$

Computing the conjunction of the individual match expressions for port intervals and single protocols is straightforward. The conjunction of two intervals in CIDR notation is either empty or the smaller of both intervals. The conjunction of two interfaces is either empty if they don't share a common prefix, otherwise it is the longest of both interfaces (non-wildcard interfaces dominate wildcard interfaces).

The type of match expressions was carefully designed such that the conjunction of *two* match expressions is only *one* match expression. If features were added to the match expression, for example negated interfaces, this would no longer be possible. Of all common features found in a firewall, we only found that it would further be possible to add TCP flags to the match expression without violating the aforementioned conjunction property.

B. Semantics of Iptables

We now outline the model of a real-world iptables firewall. Most firewall analysis is concerned with the access control rules of a firewall, therefore the model focuses on the `filter`

table. This implies, packet modification (e.g. NAT, which must not occur in this table) is not considered in this work. We rely on our previous work [3]. The model supports the following common actions: `Accept`, `Drop`, `Reject`, `Log`, `Call`ing to and `Return`ing from user-defined chains, as well as the "empty" action. The model is defined as an inductive predicate with the following syntax:

$$\Gamma, \gamma, p \vdash \langle rs,\ s\rangle \Rightarrow t$$

The ruleset of the firewall is rs and the packet under examination is p. The states s and t are in $\{\text{✓}, \text{✗}, \text{?}\}$. The starting state of the firewall is s, usually ?. The filtering decision after processing rs is t, usually ✓ or ✗. User-defined chains are stored in Γ, which corresponds to the background ruleset. A primitive matcher γ (a boolean function which takes a primitive and the packet as parameters) decides whether a certain primitive matches for a packet. Note that the model and all algorithms on top of it are proven correct for an arbitrary γ, hence, this model supports *all* iptables matching features. Obviously, there is no executable code for an arbitrary γ. However, the algorithms which transform rulesets are executable.

We make use of these algorithms, in particular: An algorithm which unfolds all calls to and returns from user-defined chains and rewriting of further actions. This leaves a ruleset where only the following actions occur: `Accept` and `Drop`. Thus, a large step for translating the real-world model to the simple firewall model is already accomplished. Translating the match expressions remains. The real-world model allows a match expression to be an arbitrary propositional logic expression. However, iptables only accepts match expressions in *negation normal form* (NNF). A Boolean formula is in NNF *iff* all occurring negations are on primitives, i.e. there are no nested negated expressions. For example, iptables can load `-s 10.0.0.0/8 ! -p tcp` but not `! (-s 10.0.0.0/8 -p tcp)`. However, such negated expressions may occur as a result of the unfolding algorithm. An algorithm to translate a ruleset to a ruleset where all match conditions are in NNF is already available [3].[1] However, there is an additional constraint imposed by iptables, not solved by the algorithm: A primitive must only occur at most once. This problem will be addressed in this paper.

We have implemented a subset of γ, namely for all primitives supported by the simple firewall and some further primitives, detailed in Section IV. Previous work provides an algorithm to abstract over all 'unknown' primitives which are not understood by our subset implementation of γ. This algorithm leads to an approximation of the ruleset. It can either be an overapproximation which results in a more permissive ruleset, or an underapproximation, which results in a stricter ruleset. For the sake of example, we will only consider the overapproximation in this paper, the underapproximation is analogous and can be found in our formalization.

Since firewalls usually accept all packets which belong to an `ESTABLISHED` connection, the interesting access control rules in a ruleset only apply to `NEW` packets. We only consider `NEW` packets, i.e. `--ctstate NEW` and `--syn` for TCP packets. Our first goal is to translate a ruleset from the real-world model to the simple model. We have proven that the set of new packets accepted by the simple firewall is a superset (overapproximation) of the packets accepted by the real-world model. This is a core contribution and we detail on the translation in the following section.

Theorem 2 (Translation to simple firewall model)**.**

$$\{p.\ \ \mathsf{new}\ p \wedge \Gamma, \gamma, p \vdash \langle rs,\ \text{?}\rangle \Rightarrow \text{✓}\}$$
$$\subseteq$$
$$\{p.\ \ \mathsf{new}\ p \wedge \mathsf{simple\text{-}fw}\ (\mathsf{translate\text{-}oapprox}\ rs) = \text{✓}\}$$

Any packet dropped by the translated, overapproximated simple firewall ruleset is guaranteed to be dropped by the real-world firewall, for arbitrary γ, Γ, rs. Similar guarantees for certainly accepted packets can be given by considering the translated underapproximation. Given the simple and carefully designed model of the simple-fw, it is much easier to write algorithms to analyze and verify the translated rulesets.

Example: We consider a `FORWARD` chain with a default policy of `DROP` and a user-defined chain `foo`.

```
-P FORWARD DROP
-A FORWARD -s 10.0.0.0/8 -j foo
-A foo ! -s 10.0.0.0/9 -j DROP
-A foo -p tcp -j ACCEPT
```

This ruleset, though it only consist of three rules and a default policy, is complicated to analyze. Our translation algorithm translates it to the simple firewall model, where the ruleset becomes remarkably simple. We use $\star$ to denote a wildcard:

```
(*,*,10.128.0.0/9,*, * ,*,*)  DROP
(*,*, 10.0.0.0/8 ,*,TCP,*,*)  ACCEPT
(*,*,      *      ,*, * ,*,*)  DROP
```

No over- or underapproximation occurred since all primitives could be translated. Note the 10.128.0.0/9 address.

IV. Translating Primitives

A firewall has the same behavior for two rulesets rs_1 and rs_2 *iff* for all packets, the firewall computes the same filtering decision for rs_1 and rs_2. Formally,

$$\forall p\ s\ t.\ \ \Gamma, \gamma, p \vdash \langle rs_1,\ s\rangle \Rightarrow t \ \ \longleftrightarrow \ \ \Gamma, \gamma, p \vdash \langle rs_2,\ s\rangle \Rightarrow t$$

In this section, we present algorithms to transform an arbitrary rs_1 to rs_2 without changing the behavior of the firewall. In the resulting rs_2, all primitives will be normalized such that the translation to the simple-fw is obvious. We continue by describing the normalization of all common primitives found in iptables rulesets.

A. IPv4 Addresses

"Modeling IP addresses efficiently is challenging." [11] First, we present a datatype to efficiently perform set operations on intervals of machine words, e.g. 32-bit integers. We will use this type for IPv4 addresses, but it can be generalized to machine words of arbitrary length, e.g. IPv6 addresses or L4 ports. We call it word interval (wi), and WI $start\ end$

[1] NNF normalizing may create additional rules.

describes the interval with $start$ and end inclusive. The Union of two wis is defined recursively.

$$\text{datatype } wi = \mathsf{WI}\ word\ word \mid \mathsf{Union}\ wi\ wi$$

Let set denote the interpretation into mathematical sets, then wi has the following semantics: $\text{set}\ (\mathsf{WI}\ start\ end) = \{start..end\}$ and $\text{set}\ \mathsf{Union}\ wi_1\ wi_2 = \text{set}\ (wi_1) \cup \text{set}\ (wi_2)$.

An IP address in CIDR notation or IP addresses specified by e.g. `-m iprange` can be translated to one WI. We have implemented and proven the common set operations: '$\cup$', '$\{\}$', '$\setminus$', '$\cap$', '$\subseteq$', and '$=$'. These operations are linear in the number of Union-constructors. The result is optimized by merging adjacent and overlapping intervals and removing empty intervals. We can also represent 'UNIV' (the universe of all IP addresses). Since most rulesets use IP addresses in CIDR notation or intervals in general, the wi datatype has proven to be very efficient. Recall that the intersection of two intervals, constructed from addresses in CIDR notation, is either empty or the smaller of both intervals.

wi is an internal representation and for the simple firewall, the result needs to be represented in CIDR notation. For this direction, one WI may correspond to several CIDR ranges. We describe an algorithm to split off one CIDR range from an arbitrary word interval r. The output is a CIDR range and r', the remainder after splitting off this CIDR range. split is implemented as follows: Let a be the lowest element in r. If this does not exist, then r corresponds to the empty set and the algorithm terminates. Otherwise, we construct the list of CIDR ranges $[a/0, a/1, ..., a/32]$. The first element in the list which is well-formed (i.e. all bits after the network prefix must be zero) and which is a subset of r is the wanted element. Note that this element always exists. It is subtracted from r to obtain r'. To convert r completely to a list of CIDR ranges, this is applied recursively until it yields no more results. This algorithm is guaranteed to terminate and the resulting list in CIDR notation corresponds to the same set of IP addresses as represented by r. Formally, $\bigcup \text{map set}\ (\mathsf{split}\ r) = \text{set}\ r$.

For example, $\mathsf{split}\ (\mathsf{WI}\ 10.0.0.0\ 10.0.0.15) = [10.0.0.0/28]$ and $\mathsf{split}\ (\mathsf{WI}\ 10.0.0.1\ 10.0.0.15) = [10.0.0.1/32, 10.0.0.2/31, 10.0.0.4/30, 10.0.0.8/29]$.

With the help of these functions, arbitrary IP address ranges can be translated to the format required by the simple firewall. The following is applied to matches on src and dst IP addresses: First, the IP match expression is translated to a word interval. If the match on an IP range is negated, we compute $\text{UNIV} \setminus wi$. All matches in one rule can be joined to a single word interval, using the $\cap$ operation. The resulting word interval is translated to a set of non-negated CIDR ranges. Using the NNF normalization, at most one match on an IP range in CIDR notation remains. We have proven that this process preserves the firewall's filtering behavior.

We conclude with a simple, synthetic worst-case example. The evaluation shows that this worst-case does not prevent successful analysis: `-m iprange --src-range 0.0.0.1-255.255.255.254`. Translated to the simple firewall, this one range blows up to 62 ranges in CIDR notation. A similar blowup may occur for negated IP ranges.

B. Conntrack State

If a packet p is matched against the stateful match condition `ESTABLISHED`, conntrack looks up p in its state table. When the firewall comes to a filtering decision for p, if the packet is not dropped and the state was `NEW`, the conntrack state table is updated such that the flow of p is now `ESTALISHED`. Similarly, other conntrack states are handled.

We present an alternative model for this behavior: Before the firewall starts processing the ruleset for p, the conntrack state table is consulted for the state of the connection of p. This state is added as a (phantom) tag to p. Therefore, ctstate can be modeled as just another header field of p. When processing the ruleset, it is not necessary to inspect the conntrack table but only the virtual state tag of the packet. After processing, the state table is updated accordingly.

We have proven that both models are equivalent. The latter model is simpler for analysis purposes since the conntrack state can be considered an ordinary packet field.[2]

In Theorem 2, we are only interested in `NEW` packets. In contrast to previous work, there is no longer the need to manually exclude `ESTABLISHED` rules from a ruleset. The alternative model allows us to consider only `NEW` packets: all state matches can be removed (by being pre-evaluated for an arbitrary `NEW` packet) from the ruleset without changing the filtering behavior of the firewall.

C. Layer 4 Ports & TCP Flags

Translating singleton ports or intervals of ports to the simple firewall is straightforward. A challenge remains for negated port ranges and the `multiport` module. However, the word interval type is also applicable to 16 bit machine words and solves these challenges. For ports, there is no need to translate an interval back to CIDR notation.[3]

Iptables can match on a set of L4 flags. To match on flags, a $mask$ selects the corresponding flags and c declares the flags which must be present. For example, the match `--syn` is a synonym for $mask$ = `SYN,RST,ACK,FIN` and c = `SYN`. For a set f of flags in a packet, matching can be formalized as $(f \cap mask) = c$. If c is not a subset of $mask$, the expression cannot match; we call this the empty match. We proved that two matches $(mask_1, c_1)$ and $(mask_2, c_2)$ are equal if and only if $(\text{if } c_1 \subseteq mask_1 \wedge c_2 \subseteq mask_2 \text{ then } c_1 = c_2 \wedge mask_1 = mask_2 \text{ else } (\neg c_1 \subseteq mask_1) \wedge (\neg c_2 \subseteq mask_2))$ holds. We also proved that the conjunction of two matches is exactly $(\text{if } c_1 \subseteq mask_1 \wedge c_2 \subseteq mask_2 \wedge mask_1 \cap mask_2 \cap c_1 = mask_1 \cap mask_2 \cap c_2 \text{ then } (mask_1 \cup mask_2, c_1 \cup c_2) \text{ else } \texttt{empty})$. If we assume `--syn` for a packet, we can remove all matches which are equal to `--syn` and add the `--syn` match as conjunction to all other matches on flags and remove empty matches. Some matches on flags may remain, e.g. `URG`, which need to be abstracted over later.

[2] This holds because the semantics does modify a packet during filtering.

[3] As a side note, OpenFlow (technically, the Open vSwitch) defines CIDR-like matching for L4 ports. With the small change of converting ports to CIDR-like notation, our simple firewall can be directly converted to OpenFlow and we have the first (almost) fully verified translation of iptables rulesets to SDN.

D. Interfaces

The simple firewall model does not support negated interfaces, e.g. `! -i eth+`. Therefore, they must be removed. We first motivate the need for abstracting over negated interfaces.

For whitelisting scenarios, one might argue, that negated interfaces is bad practice anyway. This is because new (virtual) interfaces might be added to the system at runtime and a match on negated interfaces might now also include these new interfaces. Therefore, it can be argued that negated interfaces correspond to blacklisting, which is not recommended for most firewalls. However, the main reason why negated interfaces are not supported by our model is of technical nature: Let set denote the set of interfaces that match an interface expression. For example, set `eth0` = {`eth0`} and set `eth+` is the set of all interfaces that start with the prefix `eth`. If the match on `eth+` is negated, then it matches all strings in the complement set: UNIV ∖ (set `eth+`). The simple firewall model requires that a conjunction of two primitives is again at most one primitive. This can obviously not be achieved with such sets. In addition, working with negated interfaces can cause great confusion. Note that the interface match condition '+' matches any interfaces. Also note that '+' ∈ UNIV ∖ (set `eth+`). In the second equation, '+' is not a wildcard character but the name of an interface. The confusion introduced by negated interfaces becomes more apparent when one realizes that '+' can occur as both wildcard character and normal character. Therefore, it is not possible to construct an interface match condition which matches exactly on the interface '+', because a '+' at the end of an interface match condition is interpreted as wildcard.[4]

Correlating with IP Ranges: Later, in Section V, we will compute an IP address space partition. For best clarity, this partition must not be 'polluted' with interface information. Therefore, for the partition, we will assume that no matches on interfaces occur in the ruleset. In this subsection, we describe a method to get rid of both, negated and non-negated input interfaces while preserving their relation to IP address ranges.

Interfaces are usually assigned an IP range of valid source IPs which are expected to arrive on that interface. Let $ipassmt$ be a mapping from interfaces to an IP address range. This information can be obtained by `ip route` and `ip addr`. We will write $ipassmt[i]$ to get the corresponding IP range of interface i. For the following examples, we assume

$$ipassmt = [\texttt{eth0} \mapsto \{10.8.0.0/16\}]$$

The goal is to rewrite interfaces with the corresponding IP range. For example, we would like to replace all occurrences of `-i eth0` with `-s 10.8.0.0/16`. This idea can only be sound if there are no spoofed packets; we only expect packets with a source IP of `10.8.0.0/16` to arrive at `eth0`. Once we have assured that the firewall blocks spoofed packets, we can assume in a second step that there are no spoofed accepted packets left. By default, the Linux kernel offers reverse path filtering, which blocks spoofed packet automatically. In this case we can assume that no spoofed packets occur. In some complex scenarios, reverse path filtering needs to be disabled and spoofed packets should be blocked manually with the help of the firewall ruleset. In previous work [13], we presented an algorithm to verify that a ruleset correctly blocks spoofed packets. This algorithm is integrated in our framework, proven sound, works on the same $ipassmt$ and does not need the simple firewall model (i.e. supports negated interfaces). If some interface i should accept arbitrary IP addresses (essentially not providing spoofing protection), it is possible to set $ipassmt[i]$ = UNIV. Therefore, we can verify spoofing protection according to $ipassmt$ at runtime and afterwards continue with the assumption that no spoofed packets occur.

Under the assumption that no spoofed packets occur, we will now present two algorithms to relate an input interface i to $ipassmt[i]$. Both approaches are valid for negated and non-negated interfaces. Approach one provides better results but requires stronger assumptions (which can be checked at runtime), whereas approach two is applicable without further assumptions. These approaches could be generalized to output interfaces (`-o`), which requires the routing table instead of $ipassmt$. Because a routing table may change frequently, even triggered by external malicious routing advertisements, we refrain from this rewriting in this work.

Approach One: In general, it is considered bad practice [1], [14] to have zone-spanning interfaces. Two interfaces are zone-spanning if they share a common, overlapping IP address range. Mathematically, absence of zone-spanning interfaces means that for any two interfaces in $ipassmt$, their assigned IP range must be disjoint. Our tool emits a warning if $ipassmt$ contains zone-spanning interfaces. If absence of zone-spanning interfaces is checked, then all input interfaces can be replaced by their assigned source IP address range. This preserves exactly the behavior of the firewall. The idea is that in this case a bidirectional mapping between input interfaces and source IPs exists. Interestingly, our proof does not need the assumption that $ipassmt$ maps to the complete IP universe.

Approach Two: Unfortunately, though considered bad practice, we found many zone-spanning interfaces in many real-world rulesets and hence cannot apply the previous algorithm. First, we proved that correctness of the described rewriting algorithm implies lack of zone-spanning interfaces. This leads to the conclusion that it is impossible to perform rewriting without this assumption. Therefore, we present an algorithm which adds the IP range information to the ruleset (without removing the interface match), thus constraining the match on input interfaces to their IP range. The algorithm computes the following: Whenever there is a match on an input interface i, the algorithm looks up the corresponding IP range of that interface and adds `-s` $ipassmt[i]$ to the rule. To prove correctness of this algorithm, no assumption about zone-spanning interfaces is needed, $ipassmt$ may only be defined for a subset of the interfaces, and the range of $ipassmt$ may not cover the complete IP universe. Consequently, there is no need for a user to specify $ipassmt$, but having it may yield more accurate results.

E. Abstracting Over Primitives

Some primitives cannot be translated to the simple model. Previous work already provides the function pu which removes all unknown match conditions [3]. This leads to an approximation and is the main reason for the '⊆' relation in Theorem 2. We found that we can also rewrite any known primitive *at*

[4] We greatly discourage the use of "`ip link set eth0 name +`" in production. Please fix your VM startup scripts with untrusted input now!

any time to an unknown primitive. This can be used to apply additional knowledge during preprocessing. For example, since we understand flags, we know that the following condition is false, hence rules using it can be removed: `--syn` $\wedge$ `--tcp-flags RST,ACK RST`. After this optimization, all remaining flags can be treated as unknowns and abstracted over afterwards. This allows to easily add additional knowledge and optimization strategies for further primitive match conditions without the need to adapt any algorithm which works on the simple firewall model. We proved soundness of this approach: The '$\subseteq$' relation in Theorem 2 is preserved.

V. IP Address Space Partition

In the following sections, we will work on rulesets translated to the simple-fw model. In this section, we will compute a partition of the IPv4 address space. All IP addresses in the same partition must show the same behavior w.r.t the firewall ruleset. We do not require that the partition is minimal. Therefore, the following would be a valid solution: $\{\{0\}, \{1\}, \ldots, \{255.255.255.255\}\}$. However, we will need the partition as starting point for a further algorithm and a partition of size 2^{32} is too large for this purpose. In this section, we will present an algorithm to compute a partition which behaves roughly linear in the number of rules for real-world rulesets. First, we motivate the partitioning idea with the following observation.

Lemma 1. *For an arbitrary packet p, we write $p(src \mapsto s)$ to fix the src IP address to s. Let X be the set of all src IP matches specified in rs, i.e. X is a set of CIDR ranges. If*

$$\forall A \in X.\ B \subseteq A \vee B \cap A = \{\}$$

then let $s_1 \in B$ and $s_2 \in B$ then

$$\text{simple-fw}\ rs\ p(src \mapsto s_1)\ =\ \text{simple-fw}\ rs\ p(src \mapsto s_2)$$

Reading the lemma backwards, it states that all packets with arbitrary source IPs picked from B are treated equally by the firewall. Therefore, B is a member of an IP address range partition. The condition imposed on B is that for all src CIDR ranges specified in the ruleset (called A in the lemma), B is either a subset of the range or disjoint. The lemma shows that this condition is sufficient for B, therefore we will construct an algorithm to compute B. For an arbitrary set X, this condition is purely set-theoretic and we can solve it independently from the firewall theory.

For simplicity, we use finite sets and lists interchangeably. We will write an algorithm part and reuse the common list algorithm from functional programming foldr. For X, the following algorithm computes a partition: foldr part X $\{UNIV\}$. In addition, it is guaranteed that the union of the resulting partition is equal to the universe. For our scenario, this means that the partitioning covers the complete IPv4 space. The algorithm part is implemented as follows: The first parameter is a set $S \in X$, the second parameter TS is a set of sets and corresponds to the remaining set which will be partitioned. In the first call $TS = \{UNIV\}$. For a fixed S, part S TS iterates over TS and splits the set such that the precondition of Lemma 1 holds: Written as recursive function: part S $(\{T\} \cup TS) = (S \cap T) \cup (T \setminus S) \cup (\text{part}\ (S \setminus T)\ TS)$

The result size of calling part once can be up to two times the size of TS. This means, the partition of a complete firewall ruleset is in $O(2^{|rules|})$. However, the empirical evaluation shows that the resulting size for real-world rulesets is much better. This is because IP address ranges may overlap in a ruleset, but they do not overlap in the worst possible way for all pairs of rules. Consequently, at least one of the sets $S \cap T$ or $T \setminus S$ is usually empty and can be optimized away. For example, for our largest firewall, the number of computed partitions is 10 times smaller than the number of rules. Table I confirms that the number of partitions is usually less than the number of rules.

Our algorithm fulfills the assumption of Lemma 1 for arbitrary X. Because IP addresses occur as source and destination in a ruleset, we use our partitioning algorithm where X is the set of all IPs found in the ruleset. The result is a partition where for any two IPs in the same partition, setting the src or dst of an arbitrary packet to one of the two IPs, the firewall behaves equally. This results in a stronger version of Lemma 1, which holds without any assumption and also holds for both src and dst IPs simultaneously. In addition, the partition covers the complete IPv4 address space.

VI. Service Matrices

The IP address space partition may not be minimal. That means, two different partitions may exhibit exactly the same behavior. Therefore, for manual firewall verification, these partitions may be misleading. Marmorstein elaborates on this problem [8]. ITVal's solution is to minimize the partition. We suggest to minimize the partition for a fixed service. The evaluation shows that the result is smaller and thus more clear. A fixed service corresponds to a fixed packet with arbitrary IPs. For example, we can define ssh as TCP, dport 22, arbitrary sport $\geq$ 1024. A service matrix describes the allowed accesses for a specific service over the complete IPv4 address space. It can be visualized as graph, for example Figure 1. The matrix is minimal if it cannot be compressed any further.

First, we describe when a firewall exhibits the same behavior for arbitrary source IPs s_1, s_2 and a fixed packet p:

$$\forall d.\ \text{simple-fw}\ rs\ p(src \mapsto s_1,\ dst \mapsto d) = \text{simple-fw}\ rs\ p(src \mapsto s_2,\ dst \mapsto d)$$

We say the firewall shows same behavior for a fixed service if, in addition, the analogue condition holds for destination IPs.

We present a function groupWIs, which computes the minimal partition for a fixed service. For this, the full access control matrix for inbound and outbound connections of each partition member is generated. This can be done by taking arbitrary representatives from each partition as source and destination address and executing simple-fw for the fixed packet with those fixed IPs. The matrix is minimized by merging partitions with equal rights, i.e. equal rows in the matrix. This algorithm is quadratic in the number of partitions. The evaluation shows that it scales surprisingly well, even for large rulesets, since the number of partitions is usually small.

Theorem 3 (Service Matrix is Sound and Minimal)**.** *For any two IPs in any member of* groupWIs*, the firewall shows the same behavior for a fixed service.*

For any two arbitrary members A and B in groupWIs*, if we can find two IPs in A and B respectively where the firewall shows the same behavior for a fixed service, then $A = B$.*

VII. Evaluation

We obtained real-world rulesets from over 15 firewalls. Some are central, production-critical devices. They are written by different authors, utilize a vast amount of different features and exhibit different styles and patterns. Publishing the complete rulesets itself is an important contribution (c.f. [1], [2]). To the best of our knowledge, this is the largest, publicly-available collection of real-world iptables rulesets. Note: some administrators wish to remain anonymous so we replaced their public IP addresses with public IP ranges of our institute, preserving all IP subset relationships.

Table I summarizes the evaluation's results. The first column (Fw) labels the analyzed ruleset. Column two (Rules) contains the number of rules (only the filter table) in the output of `iptables-save`. We work directly and completely on this real-world data. Column three describes the analyzed chain. Depending on the type of firewall, we either analyzed the `FORWARD` (FW) or the `INPUT` (IN) chain. For a host firewall, we analyzed IN; for a network firewall, e.g. on a gateway or router, we analyzed FW. In parentheses, we wrote the number of rules after unfolding the analyzed chain. The unfolding also features some generic, straight-forward optimizations, such as removing rules where the match expression is False. Column four (Simple rules) is the number of rules when translated to the simple firewall. In parentheses, we wrote the number of simple firewall rules when interfaces are removed. This ruleset is used subsequently to compute the partitions and service matrices. In column five (Use), we mark whether the translated simple firewall is useful. We will detail on the metric later. Column six (Parts) lists the number of IP address space partitions. For comparison, we give the number of partitions computed by ITVal in parentheses. In Column seven (ssh) and eight (http), we give the number of partitions for the service matrices for ssh and http. In column nine, we give the overall runtime of our analysis in seconds, minutes, or hours. For comparison, we put the runtime of the partitioning by ITVal in parentheses. When translating to the simple firewall, to accomplish support for arbitrary matching primitives, some approximations need to be performed. For every firewall, the first row states the overapproximation (more permissive), the second row the underapproximation (more strict).

In contrast to previous work, there is no longer the need to manually exclude certain rules from the analysis [3]. For some rulesets, we do not know the interface configuration. For others, there were zone-spanning interfaces. For these reasons, as proven in Section IV-D, in the majority of cases, we could not rewrite interfaces. This is one reason for the differences between over- and underapproximation.

We loaded all translated simple firewall rulesets (without interfaces) with `iptables-restore`. We used iptables directly to generate the firewall format required by ITVal (`iptables -L -n`). Our translation to the simple firewall is required because ITVal cannot understand the original complex rulesets and produces flawed results for them.

Fw	Rules	Chain	Simple rules	Use	Parts (ITVal)	ssh	http	Time (ITVal)
A	2784	FW (2376)	2381 (1920)	✓	246 (1)	13	9	14min (3h*)
	-	FW (2376)	2837 (581)	✗ r	1 (1)	1	1	3min (9h*)
A	4113	FW (2922)	3114 (2862)	✓	334 (2)	11	11	75min (27h*)
	-	FW (2922)	3585 (517)	✗ r	490 (1)	1	1	5min (8h)
A	4814	FW (4403)	3574 (3144)	✓	364 (2)	9	12	105min (46h*)
	-	FW (4403)	5123 (1601)	✗ r	1574 (1)	1	1	12min (3h*)
A	4946	FW (4887)	4004 (3570)	✓	371 (2)	9	12	94min (53h*)
	-	FW (4887)	5563 (1613)	✗ r	1585 (1)	1	1	11min (4h*)
B	88	FW (40)	110 (106)	✓	50 (4)	4	2	15s (2s)
	-	FW (40)	183 (75)	✓	40 (1)	1	1	9s (1s)
C	53	FW (30)	29 (12)	✓	8 (1)	1	1	7s (1s)
	-	FW (30)	27 (1)	✓	1 (1)	1	1	1s (1s)
	-	IN (49)	74 (46)	✓	38 (1)	1	1	6s (1s)
	-	IN (49)	75 (21)	✓	6 (1)	1	1	2s (1s)
D	373	FW (2649)	3482 (166)	✓	43 (1)	1	1	29s (3s)
	-	FW (2649)	16592 (1918)	✗	67 (1)	1	1	4min (33min*)
E	31	IN (24)	57 (27)	✓	4 (3)	1	2	4s (1s)
	-	IN (24)	61 (45)	✗ r	3 (1)	1	1	2s (1s)
F	263	IN (261)	263 (263)	✓	250 (3)	3	3	11min (2min)
	-	IN (261)	265 (264)	✓	250 (3)	3	3	10min (3min)
G	68	IN (28)	20 (20)	✓	8 (5)	1	2	1s (1s)
	-	IN (28)	19 (19)	✗	8 (2)	2	2	1s (1s)
H	19	FW (20)	10 (10)	✗	9 (1)	1	1	1s (1s)
	-	FW (20)	8 (8)	✗ r	3 (1)	1	1	1s (1s)
I	15	FW (5)	4 (4)	✓	4 (4)	4	4	1s (1s)
	-	FW (5)	4 (4)	✓	4 (4)	4	4	1s (1s)
J	48	FW (12)	5 (5)	✓	3 (2)	2	2	1s (1s)
	-	FW (12)	8 (2)	✓	1 (1)	1	1	1s (1s)
K	21	FW (9)	7 (6)	✓	3 (1)	1	1	1s (1s)
	-	FW (9)	4 (3)	✓	2 (1)	1	1	1s (1s)
L	27	IN (16)	19 (19)	✓	17 (3)	2	2	1s (1s)
	-	IN (16)	18 (18)	✓	17 (3)	2	2	1s (1s)
M	80	IN (92)	64 (16)	✓	2 (2)	1	2	2s (1s)
	-	IN (92)	58 (27)	✗	11 (1)	1	1	1s (1s)
N	34	FW (14)	12 (12)	✓	10 (6)	6	6	1s (2s)
	-	FW (14)	12 (12)	✓	10 (6)	6	6	1s (2s)
O	8	IN (7)	9 (9)	✓	3 (3)	1	2	1s (1s)
	-	IN (7)	8 (8)	✓	3 (3)	1	2	1s (1s)

* ITVal memory consumption, in order of appearance: 84GB, 96GB, 94GB, 95GB, 61GB, 98GB, 96GB, 21G

Table I. Summary of Evaluation on Real-World Firewalls

Performance: The code of our tool is automatically generated by Isabelle. Isabelle can translate executable algorithms to SML. For verifiable correctness, Isabelle also generates code for many datastructures which are already in the standard library of many programming languages. Usually, the machine-generated code by Isabelle can be quite inefficient. For example, lookups in Isabelle-generated dictionaries have linear lookup time, compared to constant lookup time of standard library implementations. In contrast, ITVal is highly optimized C++ code. We benchmarked our tool on a commodity i7-2620M laptop with 2 physical cores and 8 GB of RAM. In contrast, we gave ITVal a server with 16 physical Xeon E5-2650 cores and 128 GB RAM. The runtime measured for our tool is the complete translation to the two simple firewalls, computation of partitions, and the two service matrices. In contrast, the ITVal runtime only consists of computing one partition.

These benchmark settings are extremely 'unfair' for our tool. Indeed, exporting our tool to a standalone Haskell application, replacing some common datastructures with optimized ones from the Haskell std lib, enabling aggressive compiler optimization and parallelization, and running our tool on the

Xeon server, the runtime of our tool improves by orders of magnitude. Nevertheless, we chose the 'unfair' setting to demonstrate the feasibility of running fully verified code directly in a theorem prover. In addition, we preserve the property of full verification; even for the results of executable code.[5]

Table I shows that our tool outperforms ITVal for large firewalls. We added ITVal's memory requirements to the table if they exceeded 20GB. ITVal requires an infeasible amount of memory for larger rulesets while our tool can finish on commodity hardware. The overall numbers show that the runtime for our tool is sufficient for static, offline analysis, even for large real-word rulesets.

Quality of results: The main goal of ITVal is to compute a minimal partition while ours may not be minimal. Since a service matrix is more specific than a partition, a partition cannot be smaller than a service matrix. ITVal may produce spurious results (and it did in certain examples) while ours are provably correct. For firewall A, it can be seen that ITVals's results must be spurious. However, comparing the number of partitions for other rulesets, we can see that ITVal often computes better results. Our service matrices are provably minimal and can improve on ITVal's partition.

In column five, we show the usefulness of the translated simple firewall (including interfaces). We deem a firewall useful if interesting information was preserved by the approximation. Therefore, we manually inspected the rulesest and compared it to the original. For the overapproximation, we focused on preserved (non-shadowed) `DROP` rules. For the underapproximation, we focused on preserved (non-shadowed) `ACCEPT` rules. If the firewall features some rate-limiting for all packets in the beginning, the underapproximation is naturally a drop-all ruleset because the rate-limiting could apply to all packets. According to our metric, such a ruleset is of no use (but the only sound solution). We indicate this case with an $^{\mathrm{r}}$. The table indicates that, usually, at least one approximation per firewall is useful.

For brevity, we only elaborate on the most interesting rulesets and stories.

Firewall A: This firewall is the core firewall of our lab (Chair for Network Architectures and Services). It has two uplinks, interconnects several VLANs, hence, the firewall matches on more than 20 interfaces. It has around 500 direct users and one transfer network for an AS behind it. The traffic is usually several Mbit/s. The dumps are from Oct 2013, Sep 2014, May 2015, Sep 2015 and the changing number of rules indicates that it is actively managed. The firewall starts with some rate-limiting rules. Therefore, its stricter approximation assumes that the rate-limiting always applies and transforms the ruleset into a deny-all ruleset. The more permissive approximation abstracts over this rate-limiting and provides a very good approximation of the original ruleset. The ssh service matrix is visualized in Figure 1. The figure can be read as follows: The vast majority of our IP addresses are grouped into *internal* and *servers*. Servers are reachable from the outside, internal hosts are not. ip_1 and ip_2 are two individual IP addresses with special exceptions. There is also a

[5]There are methods to improve the performance and provably preserve correctness, which are out of the scope of this paper.

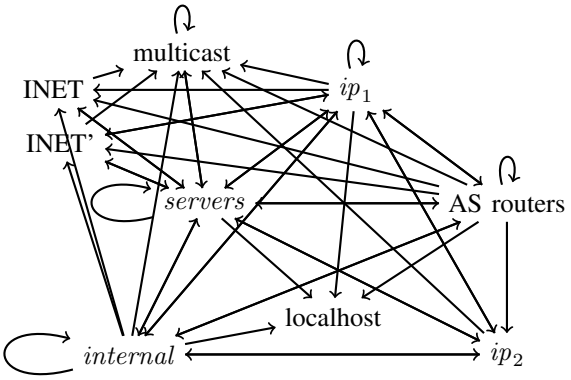


Figure 1. TUM ssh Service Matrix

group for the backbone routers of the connected AS. INET is the set of IP addresses which does not belong to us, basically the Internet. INET' is another part of the Internet. With the help of the service matrix, the administrator confirmed that the existence of INET' was an error caused by a stale rule. The misconfiguration has been fixed. Figure 1 summarizes over 4000 firewall rules and helps to easily visually verify the complex ssh setup of our firewall. The administrator was also interested in the kerberos-adm and ldap service matrices. They helped verifying the complex setup and discovered potential for ruleset cleanup.

Firewall D: This firewall was taken from a Shorewall system with 373 rules and 65 chains. It can be seen that unfolding increases the number of rules. This is due to linearizing the complex call structures generated by the user-defined chains. The transformation to the simple firewall further increases the ruleset size. This is, among others, due to rewriting several negated IP matches back to non-negated CIDR ranges and NNF normalization. However, the absolute numbers tell that this blow up is no problem for computerized analysis. The firewall basically wires interfaces together, i.e. it heavily uses `-i` and `-o`. This can be easily seen in the overapproximation. There are also many zone-spanning interfaces. As we have proven, it is impossible to rewrite interface in this case. In addition, for some interfaces, no IP ranges are specified. Hence, this ruleset is more of a link layer firewall than a network layer firewall. Consequently, the service matrices are barely of any use.

Firewall E: This ruleset was taken from a NAS device previously analyzed [3]. The ruleset first performs some rate-limiting, consequently, the underapproximation corresponds to the deny-all ruleset. In contrast to previous analysis, we obtained a more recent version of the ruleset after a system update. Our ssh service matrix reveals a misconfiguration: ssh was accidentally left enabled after the update. The service matrix for the services provided by the NAS (not listed in the table) verifies that these services are only accessible from the local network.

Firewall F: This firewall is running on a publicly accessible server. The firewall first allows everything for localhost, then blocks IP addresses which have shown malicious behavior in the past and finally allows certain services. Since most rules are devoted to blocking malicious IPs, our IP address space

partition roughly grows linear with the number of rules. The service matrices, however, reveal that there are actually only three classes of IP ranges: localhost, the blocked IPs, and all other IPs which are granted access to the services.

Firewall G: For this production server, the service matrices verified that a SQL daemon is only accessible from a local network and three explicitly-defined public IP addresses.

Firewall H: This ruleset from 2003 appears to block Kazaa filesharing traffic during working hours. In addition, a rule drops all packets with the string "X-Kazaa-User". The more permissive abstraction correctly tells that the firewall may accept all packets for all IPs (if the above conditions do not hold). Hence, the firewall is essentially abstracted to an allow-all ruleset. According to our metric, this information is not useful. However, in this scenario, this information may reveal an error in the ruleset: The firewall explicitly permits certain IP ranges, however, the default policy is `ACCEPT` and includes all these previously explicitly permitted ranges. By inspecting the structure of the firewall, we suppose that the default policy should be `DROP`. This possible misconfiguration was uncovered by the overapproximation. The underapproximation does not understand the string match on "X-Kazaa-User" in the beginning and thus corresponds to the deny-all ruleset. However, a manual inspection of the underapproximation still reveals an interesting error: The ruleset also tries to prevent MAC address spoofing for some hard-coded MAC/IP pairs. However, we could not see any drop rules for spoofed MAC addresses in the underapproximation. Indeed, the ruleset allows non-spoofed packets but forgets to drop the spoofed ones. This firewall demonstrates the worst case for our approximations: one set of accepted packets is the universe, the other is the empty set. However, manual inspection of the simplified ruleset helped revealing several errors.

VIII. Conclusion

We have demonstrated the first, fully verified, real-world applicable analysis framework for firewall rulesets. Our tool supports the Linux iptables firewall because it is widely used and well-known for its vast amount of features. It directly works on `iptables-save`. We presented an algebra on common match conditions and a method to translate complex conditions to simpler ones. Further match conditions, which are either unknown or cannot be translated, are approximated in a sound fashion. This results in a translation method for complex, real-world rulesets to a simple model. The evaluation demonstrates that, despite possible approximation, the simplified rulesets preserve the interesting aspects of the original ones.

Based on the simplified model, we presented algorithms to partition the IPv4 address space and compute service matrices. This allows summarizing and verifying the firewall in a clear manner.

The analysis is fully implemented in the Isabelle theorem prover. No additional input or knowledge of mathematics is required by the administrator. A stand-alone Haskell tool can perform the analysis automatically, only requiring the following input: `iptables-save`.

The evaluation demonstrates applicability on many real-world rulesets. For this, to the best of our knowledge, we have collected and published the largest collection of real-world iptables rulesets in academia. We demonstrated that our approach can outperform existing tools with regard to: correctness, supported match conditions, CPU time, and RAM requirements. Our tool helped to verify lack of or discover previously unknown errors in real-world, production rulesets.

Availability

The collection of firewall rulesets can be found at

https://github.com/diekmann/net-network

Our Isabelle formalization can be obtained from

https://github.com/diekmann/Iptables_Semantics

References

[1] A. Wool, "A quantitative study of firewall configuration errors," *Computer, IEEE*, vol. 37, no. 6, pp. 62–67, Jun. 2004.

[2] ——, "Trends in firewall configuration errors: Measuring the holes in swiss cheese," *Internet Computing, IEEE*, vol. 14, no. 4, pp. 58–65, Jul. 2010.

[3] C. Diekmann, L. Hupel, and G. Carle, "Semantics-preserving simplification of real-world firewall rule sets," in *Formal Methods (FM)*. Springer, Jun. 2015, pp. 195–212.

[4] L. Yuan, H. Chen, J. Mai, C.-N. Chuah, Z. Su, and P. Mohapatra, "FIREMAN: a toolkit for firewall modeling and analysis," in *Symposium on Security and Privacy*. IEEE, May 2006, pp. 199–213.

[5] V. Capretta, B. Stepien, A. Felty, and S. Matwin, "Formal correctness of conflict detection for firewalls," in *Workshop on Formal Methods in Security Engineering*. ACM, Nov. 2007, pp. 22–30.

[6] E. Al-Shaer and H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *INFOCOM*, vol. 4. IEEE, Mar. 2004, pp. 2605–2616.

[7] R. M. Marmorstein and P. Kearns, "A tool for automated iptables firewall analysis." in *USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 71–81.

[8] ——, "Firewall analysis with policy-based host classification." in *Large Installation System Administration Conference (LISA)*, vol. 6. USENIX, Dec. 2006, pp. 41–51.

[9] V. Fuller and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan," RFC 4632 (Best Current Practice), Internet Engineering Task Force, Aug. 2006.

[10] T. Nelson, A. D. Ferguson, D. Yu, R. Fonseca, and S. Krishnamurthi, "Exodus: Toward automatic migration of enterprise network configurations to SDNs," in *SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15. ACM, 2015, pp. 13:1–13:7.

[11] T. Nelson, C. Barratt, D. J. Dougherty, K. Fisler, and S. Krishnamurthi, "The Margrave tool for firewall analysis." in *Large Installation System Administration Conference (LISA)*. USENIX, Nov. 2010.

[12] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, ser. LNCS. Springer, 2002, last updated 2016, vol. 2283. [Online]. Available: http://isabelle.in.tum.de/

[13] C. Diekmann, L. Schwaighofer, and G. Carle, "Certifying spoofing-protection of firewalls," in *11th International Conference on Network and Service Management, CNSM*, Barcelona, Spain, Nov. 2015.

[14] A. Wool, "The use and usability of direction-based filtering in firewalls," *Computers & Security*, vol. 23, no. 6, pp. 459–468, 2004.

Acknowledgments

Lars Hupel commented on early drafts of this paper. We thank all (anonymous) administrators who donated their firewall configs. This work has been supported by the German Federal Ministry of Education and Research, project SURF, grant 16KIS0145, and by the European Commission, project SafeCloud, grant 653884.

Does Scale, Size, and Locality Matter? Evaluation of Collaborative BGP Security Mechanisms

Rahul Hiran
Linköping University, Sweden

Niklas Carlsson
Linköping University, Sweden

Nahid Shahmehri
Linköping University, Sweden

***Abstract*—The Border Gateway Protocol (BGP) was not designed with security in mind and is vulnerable to many attacks, including prefix/subprefix hijacks, interception attacks, and imposture attacks. Despite many protocols having been proposed to detect or prevent such attacks, no solution has been widely deployed. Yet, the effectiveness of most proposals relies on large-scale adoption and cooperation between many large Autonomous Systems (AS). In this paper we use measurement data to evaluate some promising, previously proposed techniques in cases where they are implemented by different subsets of ASes, and answer questions regarding which ASes need to collaborate, the importance of the locality and size of the participating ASes, and how many ASes are needed to achieve good efficiency when different subsets of ASes collaborate. For our evaluation we use topologies and routing information derived from real measurement data. We consider collaborative detection and prevention techniques that use (i) prefix origin information, (ii) route path updates, or (iii) passively collected round-trip time (RTT) information. Our results and answers to the above questions help determine the effectiveness of potential incremental rollouts, incentivized or required by regional legislation, for example. While there are differences between the techniques and two of the three classes see the biggest benefits when detection/prevention is performed close to the source of an attack, the results show that significant gains can be achieved even with only regional collaboration.**

I. Introduction

The Internet is highly susceptible to routing attacks [4], [12]. In almost all types of routing attacks, the attackers rely on vulnerabilities in the Border Gateway Protocol (BGP) to attract traffic that was not intended for them. Often this is achieved through a *prefix attack* or *subprefix attack*, in which the attacker announces itself as the origin of a prefix with the intention of attracting some of the traffic intended for IP addresses belonging to this prefix. Sub-prefix attacks are particularly dangerous as the longest-prefix routing rules implemented on routers always route to the most specific (sub)prefix.

An attack's severity and the complexity of detecting the attack is, to a large extent, determined by the attacker's actions when receiving the hijacked traffic. For example, *black-holing attacks* in which the traffic terminates at the attacker network are relatively easy to detect, as the traffic source may not obtain expected end-to-end responses. In contrast, *imposture attacks*, in which the attacker also impersonates the destination, or *interception attacks*, in which the attacker re-routes the traffic to the destination, are much more difficult to detect.

Unfortunately, despite an increasing number of observed routing attack occurrences [1], [6], [12], [13], it has proven difficult to incentivize operators to invest in existing solutions [4], and there is currently no universally deployed solution that prevents hijacking of Internet traffic by third parties [12]. For example, the deployment of crypto-based efforts [18], [21], [27] has been hampered by high deployment costs for network operators [4], [12]. Instead, monitoring of path announcements and the data paths taken by data packets are typically used to identify potential hijacks and other suspicious data paths [14], [16], [20], [28]. With routing paths being determined by the individual routing decisions of many involved operators and other organizations running their own Autonomous Systems (AS) [1], [16], [28], such techniques benefit greatly from information sharing between ASes.

ISBN 978-3-901882-83-8 © 2016 IFIP

TABLE I. Examples of systems, the information they share/use, and the attacks they can help detect/prevent.

Information shared	Prefix hijack	Subprefix hijack	Interception	Imposture	Example solutions
Prefix origin	✓	✓	✗	✗	Route filtering [3], [4], RPKI [21], ROVER [8]
Route path updates	✓	✓	✗	✗	PHAS [20], PrefiSec [15], PG-BGP [16]
Passive measurements	✗	✗	✓	✓	CrowdSec [14]
Active measurements	✗	✗	✓	✓	Zheng et al. [30], PrefiSec [15]

Different types of information can be helpful in the detection of routing attacks. Table I summarizes some of the most commonly proposed information sources for such systems, as well as some example systems and the types of attacks these systems propose to protect against. In this paper, we focus on the first three types and only briefly discuss the fourth type.

A number of important questions arise when considering cooperative information sharing across ASes and other network entities/organizations for the purpose of detecting or preventing routing attacks. For example, how do the detection/prevention rates of the different techniques scale with the number of participants? What is the impact of the size of each participant, or the information available to the participant? And, what is the impact of the location of the participants sharing the information? The latter question may be particularly important as it may help provide insights into the effectiveness of regional government-issued legislation or regional agreements. For example, the United States (US) government or the European Union (EU) may push to have ASes and organizations under their respective jurisdictions share information in order to protect the common interests of each region.

While some of the papers introducing the above example systems have used data-driven analysis to illustrate the power

of large-scale information sharing between large ASes, little attention has been paid to the effect of the geographic locality of each participant. Although many ASes have points-of-presence in many geographic regions, ASes operated by organizations from the same country or geographic region may be more likely to openly or through legislation, for example, share information with each other. Ongoing geographical and political polarization may further contribute to potential location-based participation and sharing restrictions. Motivated by these observations, in addition to analyzing each of the above three questions on their own, this paper places particular focus on the impact of the locality of the participants. Locality is considered both on its own, and also with regards to size-based inclusion within and across regions, as well as with regards to the scale of the (local or global) information sharing alliance.

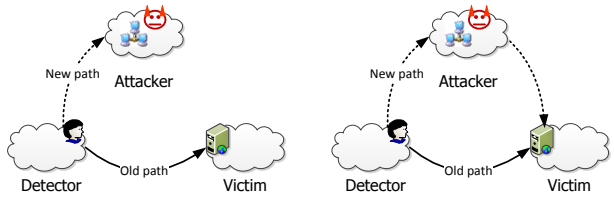
The main contribution of this paper is a systematic data-driven evaluation of some promising-previously proposed hijack prevention and routing attack detection techniques. In particular, we consider the above outlined questions in the context of three example techniques that share (i) prefix origin information, (ii) route path updates, or (iii) passively collected round-trip time (RTT) information. For our evaluation, we develop a data-driven methodology for each information sharing approach which takes into account the geographic locality (e.g., the region in which the AS is registered) and the relative size (e.g., measured by the number of neighboring ASes) of each of the potential participants. Using real-world topologies and routing information derived from measurement data we then systematically evaluate the impact of each factor, either on its own, or accounting for the geographic locality of the participants, attackers, and victims.

Our results provide insights into the tradeoffs between global and local deployment. While the results highlight the value of detection and prevention close to the source of an attack, we also find cases where regional collaboration may achieve many of the benefits achievable through global deployment. Other interesting findings include the observation that the largest ASes are not always the best at hijack detection when the attacks are from other regions. Instead, collaboration with mid-sized ASes may be beneficial. This is in contrast to the deployment of hijack prevention mechanisms, which benefit significantly from large ASes participating, regardless of whether the deployment is global or regional. Our scale- and size-based evaluation also provides insights into other deployment related issues, including the relative deployment benefits during different phases of an incremental rollout.

Paper outline: Section II provides background and sets the context. The following three sections present our evaluation results for three general classes of collaborative prevention and detection systems. In Section III we evaluate (sub)prefix attack prevention techniques that use prefix origin information, in Section IV we evaluate hijack detection mechanisms that use path announcements, and in Section V we evaluate interception and imposture detection techniques that use passively collected RTT measurements. Finally, related work and conclusions are presented in Sections VI and VII, respectively.

II. Background

BGP works well in normal circumstances. However, inherent vulnerabilities with the protocol enable routing attacks.



(a) Imposture attack (b) Interception attack

Fig. 1. Imposture and interception examples.

A. Routing attacks

A major vulnerability in BGP is its inability to validate the allocation of prefixes to ASes. This makes it difficult to detect when an attacker AS announces one or more prefixes allocated to other network(s). In a *prefix hijack* the attacker announces a prefix (e.g., a.b.c.d/16) that is actually allocated to a different AS. Depending on ASes' relationships and how the AS-PATH is propagated between ASes such attacks may attract (hijack) more or less traffic. In a *subprefix hijack* the attacker announces a subprefix (e.g., a.b.c.d/24) of a larger prefix (e.g., a.b.c.d/16). Due to the longest-prefix matching rule used by routers, these attacks may be particularly effective in hijacking traffic.

All the above types of attack may lead to one of several outcomes. For example, in a *blackholing attack* the attacker simply drops the traffic that it attracts. Figure 1 illustrates two more difficult attacks to detect. In an *imposture attack* (Figure 1(a)), the attacker impersonates the intended destination for the traffic and in an *interception attack* (Figure 1(b)) the attacker redirects the traffic to its intended destination, possibly after making a copy or modifying the data, for example. These attacks are particularly stealthy when the users originating the traffic receive uninterrupted service.

B. Collaborative information sharing

Various systems have been proposed to detect, mitigate, or prevent routing attacks and other unwanted routing incidents. These systems typically rely on collaborating ASes sharing different information. For our analysis we focus on three broad classes of techniques that share and/or use the first three types of information in Table I. They correspond to prefix origin information, route path updates, and passively collected RTTs. Route path updates can easily be collected at individual routers or at the AS level, and then shared with other ASes. RTTs can easily be passively collected and shared by almost any network entity [14]. In the following we describe how the different systems that we evaluate here use the shared information.

C. Hijack prevention using prefix origin

Ideally, a hijack prevention mechanism should prevent an AS from accepting and propagating bogus route announcements. If implemented widely, such mechanisms could then prune away bogus route announcements close to the source and prevent an attacker from reaching more ASes and users.

Previously proposed mechanisms that can provide hijack prevention include prefix filtering [4], PG-BGP [16], RPKI [8], and ROVER [17]. Many of these techniques build a trusted and formally verifiable database of prefix-to-AS pairings between the IP prefixes and the ASes that are allowed to originate them. If used correctly, routers implementing RPKI [8] and

ROVER [17] can be assumed to pick the right origin and avoid propagating the wrong origin for prefixes. To achieve this, two conditions must be satisfied. First, the AS that wants to defend itself against route hijacks on its prefixes, or its provider AS, must register a prefix-to-AS mapping with RPKI or ROVER. Second, as part of their validation of prefix-to-AS mappings, relying ASes must successfully retrieve and check these AS-to-prefix mappings from the RPKI or ROVER records.

With PG-BGP [16], the acceptance of suspicious routes is delayed, and routes are accepted and propagated only after a certain threshold time duration has passed. Since suspicious routes are typically short-lived [25], the performance of PG-BGP is usually similar to that of RPKI and ROVER. Therefore, for the purpose of our evaluation, we only simulate the performance of RPKI and ROVER.

D. Control-plane based anomaly detection

There are several works that are based on control-plane data for the detection of anomalies in BGP routing, including PHAS [20], PrefiSec [15], and PG-BGP [16]. While PHAS and PG-BGP aggregate all information centrally, PrefiSec distributes computing and detection across participants. Otherwise, the approaches are relatively similar. For each prefix, these protocols track the origin ASes observed by its participants and raises alerts when there are changes. The common idea leveraged by all these protocols is that an IP prefix should be originated by a single AS. An IP prefix originated by more than one AS results in a Multiple Origin AS (MOAS) conflict. While some MOAS are legitimate and can be observed over long time periods [29], a newly-detected MOAS conflict can be an indication of a potential prefix hijack. By keeping track of the AS-to-prefix mappings observed in AS-PATH announcements, these protocols can flag new potential MOAS cases. Naturally, as more ASes participate and share their observed path announcements, the system will have more complete AS-to-prefix mappings.

E. Route anomaly detection using passive measurements

The examples in Figure 1 illustrate why imposture and interception attacks may be particularly difficult to detect without observing the actual data path or the impact these changes have on the RTTs. Both active traceroute-based anomaly detection techniques [15], [30] and passive RTT-based anomaly detection techniques [14] have been proposed. While we will focus on the use of passive measurements, we note that the approaches in general are fairly similar. For example, Zheng et al. [30] use changes in the number of hops in the traceroute paths to identify potential hijacks, while Hiran et al. [14] use changes in the RTTs to identify potential anomalies.

In both types of systems measurement information from multiple sources is shared to provide stronger evidence and more accurate flagging of suspicious events. For example, in CrowdSec [14] clients or other network entities (e.g., middleboxes) share RTT outlier information and collaboratively identify prefixes with many affected clients, so as to identify potential routing anomalies. For collaborative detection the system uses statistical tests based on binomial hypothesis testing. One of the main advantages of using passive measurements is that, in contrast to active measurements such as traceroutes, they do not add additional traffic overhead.

III. Evaluating hijack prevention techniques

Several studies have suggested that there are significant benefits to deploying hijack detection and prevention mechanisms on several large ASes across the world. However, global deployment that spans multiple geographic regions and jurisdictions is non-trivial and may not be practical due to political and economic reasons. It may be more practical to push or incentivize the deployment within a geographic region such as the US or EU. For example, governmental legislation or other regional mechanisms may be used to push or incentivize agreements between ASes within a region.

In this paper, we evaluate and compare the benefits and drawbacks of deploying three different general classes of prevention and detection techniques regionally versus globally. For each class of techniques we simulate the effectiveness of the general technique when different subsets of potential candidate participants employ the technique and share information between each other. Within this context, we then answer questions related to the impact of locality and size of the participants, as well as the number of participants. For example, what is the impact of the number of ASes that deploy the hijack prevention mechanisms, either from a specific region (e.g., North America or Europe) or globally? And, what is the impact of size of the ASes that deploy the hijack prevention mechanisms from a specific region or globally?

In this section we answer the above questions in the context of hijack prevention mechanisms such as route filtering [3], [4], RPKI [21], and ROVER [8].

A. Simulation-based Evaluation Methodology

For our simulation-based evaluation, we modified and extended the existing BSIM [16] simulator. BSIM simulates route propagation using the standard Gao-Rexford model [7], which captures the behavior of the economy-driven policies used in practice [11]. The model distinguishes between *customer-provider* relationships (where the customer AS pays its provider) and *peer-peer* relationships (where two ASes often agree to transit each other's traffic for free). In particular, the model assumes that ASes use a routing policy in which customer routes may be exported to all neighboring ASes, but routes learned from peers or providers are exported only to the customers. In addition, the policy prefers customer routes over peer routes (since they bring revenue) and peer routes over provider routes (since provider routes cost money). In cases of multiple tied routes, the routes with the shortest AS paths are chosen. Finally, for the purpose of the simulations, if there are still ties, these ties are broken (arbitrarily) by picking the route over the AS with the lowest AS number.

We extracted the Internet AS-level topology and AS relationship information for every pair of neighboring ASes from public data [5]. We use a snapshot from August 2015, which contains 51,507 ASes and 199,540 AS relationships.

For evaluating hijack prevention mechanisms, we simulate how the routes would propagate in the presence of hijack prevention mechanisms compared to the case when these mechanisms are not present. We measure the fraction of ASes that end up forwarding packets along the correct path in both scenarios and report the percentage increase in the number of

ASes that choose the correct origin. To calculate the percentage increase we first simulate each example attack when no ASes deploy the prevention mechanism and when a random subset of ASes deploy the mechanism, respectively, and then report the average increase in the number of ASes that route to the correct destinations when the prevention mechanism was deployed. In each example scenario, we perform simulations by randomly choosing victim and attacker ASes from selected example regions. Across all scenarios, we randomly picked N ASes to deploy the mechanism from the set of ASes with at least X neighboring ASes, and reported the average over 500 simulations per scenario (with 95% confidence intervals for the average). The use of threshold is in part motivated by larger ASes (with many neighbors) being more likely to have the resources to deploy hijack prevention mechanisms [10]. Here, the degree threshold X is used to bias the size of the individual participants and the parameter N captures the size (scale) of the alliance as a whole.

To compare different deployments, we use locality-, size-, and scale-based criteria to randomly pick subsets of the nodes on which to implement the mechanism. In all simulations, victim nodes are selected at random and the reported metrics are calculated over all nodes in the network.

As is common practice, for our evaluation we varied one parameter at a time, while keeping all the other parameters constant. Our default degree threshold $X = 20$ was selected to map to an intermediate value in the range of interest (0-50), and the default alliance size N was selected to be equal to the number of ASes with at least 50 neighbors.

Before presenting our results, it should be noted that the simulations have limitations. First, the AS relationship data used for the simulations is not perfect and does not take into account more complex AS-to-AS relationships. For example, two ISPs may interconnect at multiple peering points and have different types of relationships at each point [5]. Second, not all network operators follow the standard rules for route export. However, it is believed that there are few exceptions [11].

B. Global Baseline: Scale and Size

For reference, we first present results when the participating ASes are selected from the global set of ASes. Figure 2 summarizes these results. Figure 2(a) shows the percentage improvement in the number of ASes that chose the correct origin, as a function of the number of participating ASes. With our default threshold $X = 20$, the right-most points correspond to the case when all 2,626 ASes with at least 20 neighbors participate. In comparison, Figure 3 shows the same plot for ASes in North America (NA), the European Union (EU), and the rest of the world (all ASes excluding those in NA and EU).

Referring to the global deployment results (Figure 2(a)), all regions observe significant advantages from higher participation. For example, with 500 random participants we observe an average improvement of more than 15% across all victim-attacker pair scenarios. In comparison, hen all ASes with degree of at least 20 participate the improvements are consistently above 45%. While overall numbers are lower when only local ASes participate (Figure 3), we note that local deployment is important when protecting against attacks from within the region. This is demonstrated by the higher percentage of improvements when the attacker is in the region deploying the security mechanism.

Figure 2(b) shows the percentage improvement as a function of the threshold degree X. With our default alliance size $N = 1,093$, the righ-most points correspond to the case when all ASes with a degree of at least 50 participate.

From these results it is clear that the high-degree ASes are the ones that offer the most protection. For example, if all the 1,093 top-ASes with more than 50 neighbors participate we observe improvements of more than 40% for all victim-attacker scenarios. This shows the importance of getting the large ASes onboard in these deployment efforts. The general observation that collaboration by a few large ASes can provide much of the protection is not new. Similar observations have been made by Gersch et al. [9] and Karlin et al. [16], for example. In this work, we take this analysis one step further and consider the impact of regional deployment.

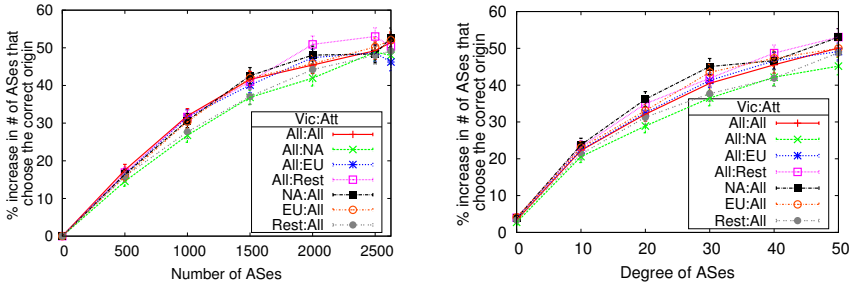
Figure 4 presents location-based results for when only ASes in a certain region deploy the prevention mechanism, and where ASes deploying the mechanism are selected based on their degree. We note that regional deployment can provide similar improvements as in a global deployment when the attacker is local. The improvements are noticeably lower when the attackers are located in other regions. For example, the percentage gain in ASes choosing the correct origin for attackers in NA is greater when ASes in NA deploy the prevention mechanisms compared to the gains when ASes in other regions deploy the mechanisms. These results illustrate that enforced deployment of these mechanisms may be a good way for regions to clean up their own networks.

The locations of victim networks play a smaller role. Although all networks would benefit from such a deployment, the local networks would not gain much more protection than external networks. These mechanisms should perhaps best be seen as mechanisms for the greater good, with the results showing that there is great incentive for governments and network operators to come together to help ensure that prevention mechanisms are deployed on a large scale.

C. Location-based Discussion: Key Findings

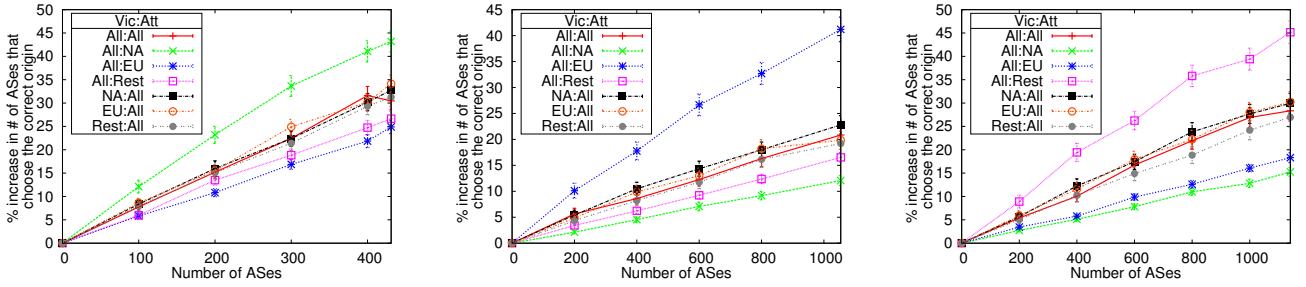
It is often stated that you should *keep your friends close and your enemies closer*. Our results highlight that this is also an important lesson in today's networks. First, starting with our friends, our results show that there are substantial gains from local deployment, regardless of where the attacks come from. For example, if all ASes with a degree of at least 20 deploy these mechanisms we observe a 30% gain for the NA-based victims regardless of attacker region (e.g., Figure 3(a)). In EU (Figure 3(b)) the corresponding gain is 20%.

Second, considering the attackers, our regional results clearly show that detectors close to the attackers help the most. For example, when ASes in NA deploy hijack prevention schemes, the damage from hijack attacks originating from ASes in this region can be significantly controlled in all regions. These results show that the largest benefits come with global deployment, and that there may be benefits to subsidizing or otherwise incentivizing international partners to implement these mechanisms too.



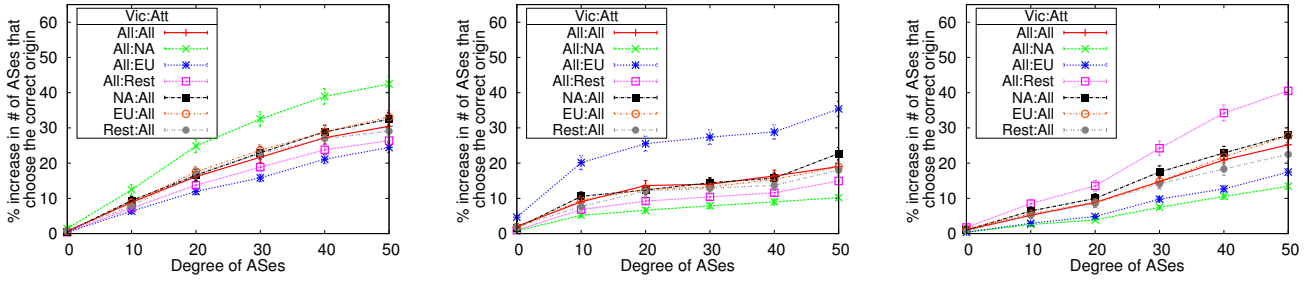
(a) Number of participating ASes (b) Size of participating ASes

Fig. 2. The average percentage improvement in the number of ASes that choose the correct origin when different subsets of the global set of ASes participate.



(a) North America (NA) (b) European Union (EU) (c) Rest of the world

Fig. 3. Impact of number of participating ASes, when ASes are selected from a particular geographical region or the "rest of the world".



(a) North America (NA) (b) European Union (EU) (c) Rest of the world

Fig. 4. Impact of the degree threshold of the participating ASes, when all are selected from a geographic region or the "rest of the world". For these figures, we choose $N = 207$, $N = 571$, and $N = 315$, respectively.

Overall, the percentage gain when hijack prevention mechanisms are deployed by all (roughly 2,500) ASes around the world with a degree of at least 20 varies between 40% and 50% for different combinations of victim-attacker regions (Figure 2(a)). If only 500 random ASes around the world deploy the mechanism, the gain is roughly 15%. By comparison, when the 431 ASes with a degree of at least 20 in NA deploy the hijack prevention mechanisms, the percentage gain varies between 23% to 43%, depending on which victim-attacker pair combination is considered (Figure 3(a)). The higher numbers partially reflect the big impact of the NA-based ASes, many of which are high-degree ASes with peering points around the globe, but also demonstrate the value of regional deployment to help protect against hijack attacks.

IV. Evaluating Hijack Detection Mechanisms

A. Methodology and Datasets

To evaluate hijack detection mechanisms based on AS-PATH updates we have extended and modified a framework that was previously used to evaluate alert rates for PrefiSec [15] to account for ASes and their locality. While the evaluation framework is designed for PrefiSec, the results presented here also apply to PHAS [20] and PG-BGP [16]. Given the same information, these systems' detection rates are the same. The main differences between these systems are their communication overhead and where the processing is performed.

For our analysis, we collected the RIB files and AS-PATH announcements observed at all six routeviews servers active during the time of the China Telecom incident [13], on April 8, 2010, when China Telecom announced origin for 50,000 prefixes originated by other ASes. Using announcements from around the time, we compare differences and similarities of the detection rates during an actual attack.

Focusing on a two-week window around the time of the incident, we first used the RIB files from April 1 and a warm-up period to initialize the AS-to-prefix mappings seen by different selected subsets of ASes. Of particular interest here is the degree (size) and locality of the collaborating ASes. In our evaluation, we consider sets of collaborating ASes selected from NA, EU, the "rest of the world" (reference point, rather than a region), and from the global set. In contrast to the original evaluation frameworks, which treated the routeviews servers as the participants [15], we use the AS information of the ASes contributing announcements to the routeviews servers and AS-to-region mappings to identify subsets of information seen by different subsets of collaborating ASes.

In total, the six routeviews servers have 100 vantage points that belong to 73 unique ASes. Of these, 38 are NA-based, 21 EU-based, and 14 map to other geographic regions.

For each of the subsets of collaborating ASes that we chose, we then look at each day in the time window and simulate and report the number of prefixes and origins, respectively, that the ASes in the subset would not have seen prior to that day. These two metrics directly measure the number of cases that must be flagged (and further investigated) as potential prefix and sub-prefix attacks, respectively.

B. Global Baseline

As a baseline, we first present results for when the collaborating ASes are selected globally. Figure 5(a) shows the number of alerts raised for both "new prefixes" (possible subprefix hijacks) and "new prefix origins" (possible prefix hijacks) announced during the incident (on April 8) as a function of number of collaborating ASes. We also include separate lines for the number of alerts of these two types raised due to announcements made by China Telecom.

We see that the number of alerts for possible prefix hijacks increases with the number of collaborating ASes, and that 40,575 alerts (for both prefix and subprefix hijacks) are raised during the day of the attack if all the nodes collaborate. With the exception of a few "new prefixes" and "new prefix origins", almost all alerts are due to the China Telecom announcements associated with the incident, which caused traffic for these prefixes and subprefixes to be hijacked.

Only a few ASes are needed to detect the majority of the subprefix hijacks ("new prefixes"). This result can be explained by subprefixes being propagated to almost all ASes due to more specific prefixes being preferred. For prefix attacks ("new origin") additional ASes are much more beneficial, with some diminishing returns after reaching 40 ASes. This happens because ASes during these instances become divided into two groups: ASes that continue routing to the victim network and ASes that choose to route to the attacker network. Thus, additional collaborating ASes increases the chance that conflicting origins are detected and hijack alerts are raised.

Figure 6 puts the above numbers in perspective, showing the number of alerts for the days before and after the attack. In addition to being orders of magnitude lower than during the day of the incident, the flatter "new origin" curves suggest that the "new origin" announcements during these days propagated somewhat further than the China Telecom announcements.

Figure 5(b) shows the number of alerts as a function of the degree threshold to be included in the alliance. For every threshold, 10 ASes with a degree of at least X are selected at random. Here, the right-most displayed threshold is picked so that the selection set include exactly 10 ASes, and the following points (moving to the left) are picked so as to roughly double the selection set for each point. The degree threshold of 1 is included as a reference point.

The figure shows that the number of alerts for the China Telecom incident is higher when the degree threshold is small, and the number of alerts is quite low when large ASes collaborate. This is a very interesting observation as much prior work has suggested collaboration between the largest ASes, but it can be partially[1] explained by most of the high degree ASes being NA-based. For example, of the ASes with a degree greater than 1,174, all but one (i.e., 9 of 10) are NA-based, and when the threshold is 646, there are 18 NA-based and 2 EU-based. However, these NA-based ASes do not have as good a vantage point of the China-based incident, with only a subset of the paths propagating to these ASes. With a lower degree threshold more ASes from outside NA and EU will be included, improving the results. This illustrates that the vantage points offered by global collaboration can be more valuable to the prefix hijack detection than having only the large ASes collaborate. Similarly, multi-hop BGP peering can also help. The detection numbers for subprefix attacks ("new prefixes") are less dependent of the AS degree (size) and locality; again, indicating their wider propagation.

C. Location-based Analysis

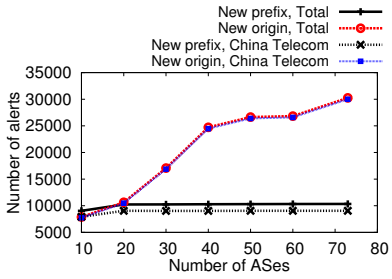
We now discuss the benefits of regional collaboration for hijack detection. Figure 7 shows the number of alerts as a function of number of ASes for different regions. For all of the three regions (NA, EU, and "rest of the world"), the number of alerts increases as more ASes share information. If all NA-based ASes collaborate there are 22,178 alerts (13,214 "new origin" and 8,964 "new prefix"). Sharing among all EU-based ASes raises 10,829 (3,620+7,209) alerts and sharing among all the ASes in the "rest of the world" category would raise 36,328 (27,280+9,048) alerts. Whereas the sub-prefix detection ("new prefix") is similar for the different regions, the differences in total alerts are substantial. For example, despite there being far fewer ASes in the "rest of the world" category, this category has the highest detection rate. The main reason for this is that many of these ASes have more vantage points closer to China Telecom than NA-based and EU-based ASes may have, and therefore have better visibility of the route announcements made by China Telecom. This observation mirrors the insights provided by our hijack prevention results (Section III) that show that ASes deploying protection mechanisms close to the attacker provide the best protection.

While none of the regional collaborations performs as good as global collaboration, the value of regionally deployed solutions should not be underestimated, especially as there is no solution that has seen widespread deployment yet. These results show that careful regional deployment, possibly with a few complementing ASes from other regions, may provide a significant step in the right direction.

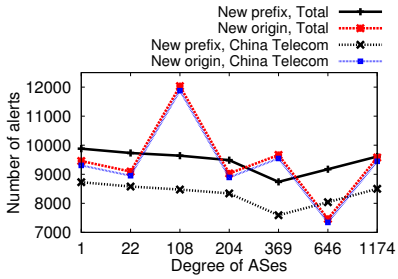
Figures 8(a) and 8(b) show the number of alerts as a function of the degree threshold for regional collaborations in NA and EU, respectively. As for the global results, for each degree threshold, we randomly pick 10 ASes per alliance.

We again observe stronger degree (size) dependence for prefix hijack detection ("new origins") than for subprefix hijack detection ("new prefixes"). While the large ASes in NA in general provide more alerts than the smallest ASes in NA, it is very interesting that the very top ASes see a drop in the number of alerts they raise. It is also interesting that the large ASes in EU detect fewer attacks than the smaller ASes in EU. As the above ASes are in the same region, our previous explanations (in Section IV-B) regarding the relative differences in coverage

[1] Additional explanation will be provided in the next subsection.



(a) Number of participating ASes



(b) Size of participating ASes

Fig. 5. Average number of alerts raised when global ASes collaborate the day of the China Telecom incident.

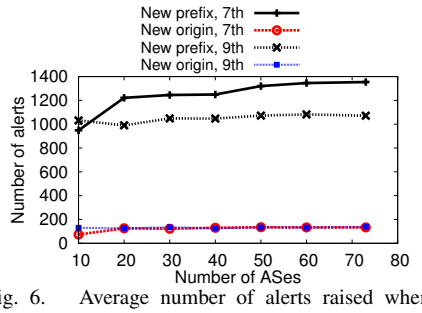
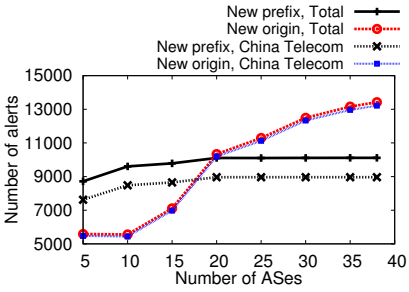
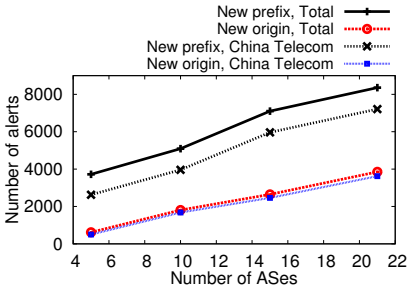


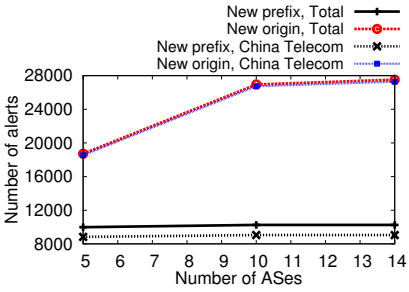
Fig. 6. Average number of alerts raised when global ASes collaborate the day before (April 7) and after (April 9) the incident.



(a) North America (NA)



(b) European Union (EU)



(c) Rest of the world

Fig. 7. Number of alerts during the day of the incident (April 8, 2010) for different sizes of regional collaborations.

seen by ASes in *different regions* no longer apply here. In the *same region*, the size-based differences may instead be related to the standard route export policy. In particular, malicious routes (learnt from a peer or provider) are typically exported only to customers. Therefore, malicious routes learnt by mid-tier ASes may not reach their providers (typically large ASes).

V. Interception and Imposture Detection

A. Methodology and Datasets

To provide insights into the impact of regional collaboration for detecting interception and imposture attacks, we have extended the evaluation of CrowdSec [14] to account for locality of the collaborating network entities. CrowdSec is designed to raise alerts about RTT anomalies and help detect interception and imposture attacks. In the case of an interception attack (Figure 1(b)) the RTTs typically increase during an attack, whereas the RTTs during an imposture attack (Figure 1(a)) can either increase or decrease, depending on the relative locality of the attacker, victim, and detector.

In CrowdSec the end clients passively collect RTT measurements while in contact with different candidate victim IP addresses (or prefixes). The client applies an outlier detection test to raise an alert if the new RTT measurement deviates significantly from previously observed RTT measurements. These alerts are shared with other CrowdSec clients, and the individual alerts are combined using statistical test methods such as a binomial test that takes into account the likelihood of N clients observing significant deviations in RTT measurements to the same prefix, given past observations [14].

For the evaluation presented here, passively collected RTT values are simulated by extracting RTTs from (active) traceroute measurements performed by PlanetLab[2] nodes as part of the iPlane [23] project. In particular, we use daily RTT measurements associated with 106 NA-based nodes, 79 EU-based nodes, and 36 nodes located in other parts of the world. For the most part we use a month's worth of training data (e.g., 278,690 successful traceroutes during July 2014) and evaluate the performance of different detection techniques for the following week, during which we simulate different attack combinations. In total, we simulate 15,279 interception attacks and 62,576 imposture attacks per set of sample detectors, and report results averaged over 10 such sample sets. While we only present interception results, the results for imposture attacks are similar. In each simulation, detector nodes and affected nodes are selected randomly within each region.

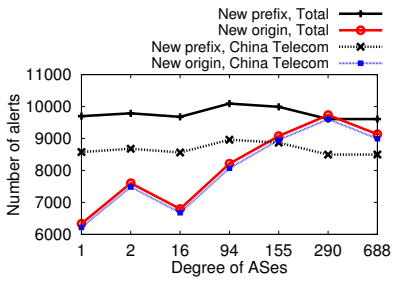
[2]PlanetLab, https://www.planet-lab.org/

B. Global and location-based evaluation

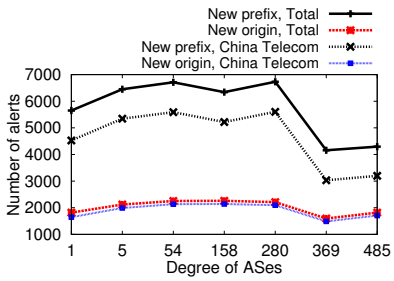
Figure 9 shows a comparison of tradeoffs in detection rate during a simulated attack (y-axis) and the false alert rate under normal circumstances (x-axis), when all global vantage points are collaborating (Global) and when only those in North America (NA) or Europe (EU) collaborate. We include results for when all (100%), half (50%), or none (0%) of the potential detector nodes are affected. The case when no nodes (0%) are affected is included only as a reference point, and captures the false positive rates during normal circumstances.

Interestingly, the 106 NA-based nodes achieve a tradeoff that is almost as good as the larger global collaboration (with 221 nodes). For example, a global collaboration that allowed a false alert rate of 10^{-2} would achieve a detection rate of 80%, if 50% of the nodes were affected. In the same scenario, the NA-based nodes achieve a 70% detection rate and the EU-based nodes achieve a 40% detection rate.

Note, however, that the size of the collaboration may play a big role. In Figure 10 we present a regional comparison while keeping the number of detector nodes fixed at 20 and



(a) North America (NA)



(b) European Union (EU)

Fig. 8. Impact of the size of the participating ASes on the number of alerts. For each degree threshold we choose 10 ASes with a degree equal to or greater than the applied threshold.

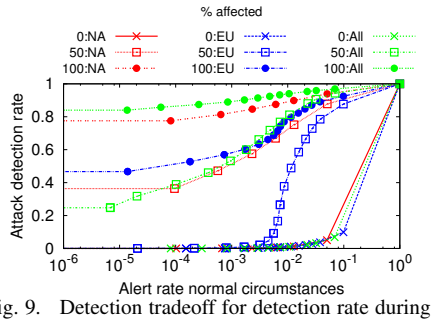


Fig. 9. Detection tradeoff for detection rate during attack and false alert rate during normal circumstances for varying percentages of affected nodes.

30. It turns out that NA-based nodes provide much better detection than EU-based nodes, even when taking alliance size into account, and in fact outperform a global alliance. Part of the reason for these differences may be differences in the variability of RTTs. Another possible contributing factor that we have discovered is that some EU-based routes (even between two EU-based nodes) go through NA even under "normal" circumstances. In such cases, attacks by networks outside EU may not result in noticeable changes in the RTTs.

C. Scale of collaboration

In general, regardless of the locality of the alliance, we have observed significant advantages to larger alliances. This is illustrated in Figure 11. Here, we show the alert-rate tradeoff for collaborations of different sizes when including nodes that are randomly selected from all global nodes (Figures 11(a)) vs. only North America (Figures 11(b)). Related to scale, it should also be noted that there are benefits to larger numbers of RTT measurements, as this helps to filter out anomalies. While region-based analysis of this aspect is omitted here, we refer the interested reader to our global results [14].

VI. Related Work

A large number of security mechanisms have been proposed to secure Internet routing. As described in Section II, this includes prefix hijack prevention mechanisms based on prefix filtering [3], [4], crypto-based solutions such as RPKI [21] and ROVER [8], hijack detection mechanisms based on changes in prefix origins observed in AS-PATH announcements (e.g., PHAS [20], PrefiSec [15], and PG-BGP [16]), and route hijack detection mechanisms using either passive RTT measurements [14] or active traceroute measurements [15], [30]. Rather than proposing new mechanisms, we evaluate the effectiveness of three broad classes of such mechanisms when they are only partially deployed. We place particular focus on the geographic locality of the collaborating ASes or network entities, while also considering the impact of the collaboration scale and the size of participating ASes.

While partial deployment of BGP security mechanisms has been considered in prior literature [2], [10], [16], [22], the geographic location of participants is almost always ignored. Instead, carefully selected ASes have typically been used to demonstrate the potential of the individual techniques. For example, Avramopoulos et al. [2] demonstrate good protection of a participant's outgoing and incoming traffic using only the top-5 tier-1 ASes in the world. Others have relied on the top-tier ASes to demonstrate the effectiveness of PG-BGP [16], path validation protocols such as S-BGP and BGPSec [22], and incentive strategies for deployment of S*BGP [10]. None of these works consider the impact of locality of the ASes that are deploying the security mechanisms.

We are not the first to study the impact of the number of participating ASes [9] or their node degree [24]. For example, Suchara et al. [24] analyze security gains as a function of increasing the node degree of the ASes that use a BGP security mechanism that filters malicious routes. Similarly to our results, they find significant benefits to deploying the mechanism at high-degree ASes at the core of the Internet. Gersch et al. [9] analyze the effect of increasing the number of ASes using attack prevention techniques. Their results nicely show how the average number of polluted ASes decreases as the number of participating ASes (with higher degrees) increases. Again, none of these works consider which geographic region each AS maps to. This can be an important factor when it comes to legislation and other political incentives.

Much work has also been done to understand the slow adoption of RPKI and other solutions [10], [26]. Other orthogonal but interesting work in this domain has designed AS reputation systems that use control-plane information to capture short-lived routes often used by malicious ASes [19].

Finally, the original simulation framework used in Section III has also been used by Karlin et al. [16]. For this part, we extend the simulator to take into account the geographic locations of the attackers, victims, and collaborating participants. In Sections IV and V we extend and generalize our prior evaluation frameworks for PrefiSec [15] and CrowdSec [14]. Again, neither of these systematically evaluates the value of scale and size in the context of locality-restricted collaboration. This paper evaluates three such broad classes of mechanisms.

VII. Conclusions

Despite BGP's vulnerabilities and increasingly many routing attacks, no universally deployed security solution to such attacks exists. Using simulations based on real measurement data we have presented a systematic evaluation of three broad classes ofprevention and detection techniques. We have focused on the impact that regional, rather than global, deployment could have on their ability to prevent/detect attacks, as well as the impact of AS size of the (regional or global) participants and the number of ASes that deploy the techniques. While prefix hijack prevention (Section III) and detection (Section IV) benefit greatly from deployment close to the source of an attack, it is encouraging to see cases with

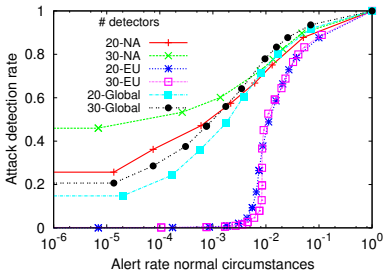


Fig. 10. Detection tradeoff, when keeping the number of detectors fixed. Here, 50% of the nodes are assumed to be affected.

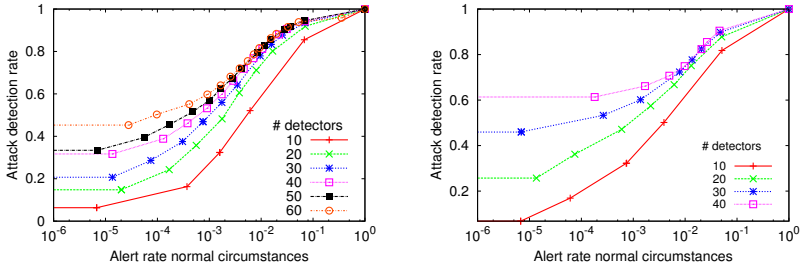


Fig. 11. Detection tradeoff for detection rate during attack and false alert rate during normal circumstances for varying numbers of detectors. Here, 50% of the nodes are assumed to be affected.

all three classes of techniques where regional deployment provides substantial benefits. We even find some cases where regional deployment achieves most of the benefits achievable through global deployment, and note that regional deployment with carefully selected participants (e.g., based on AS size) can outperform global deployments that are less carefully planned. Another interesting observation is that the largest ASes can provide worse detection than mid-sized ASes, which may see a richer set of bogus announcements. This contrasts to deploying hijack prevention mechanisms, for which large ASes appear to provide the greatest benefit. The best AS selection may therefore depend on if the system is designed for prevention or detection. We have focused on one class of techniques at a time. Interesting future work could weigh the benefits of the different approaches against each other for different collaboration constellations.

Acknowledgements

The authors are thankful to our shepherd Ignacio Castro and the anonymous reviewers for their feedback, which helped improve the clarity of the paper.

References

[1] Arnbak, A., and Goldberg, S. Loopholes for circumventing the constitution: Unrestrained bulk surveillance on Americans by collecting network traffic abroad. In *Proc. HOTPETS* (Jul. 2014).

[2] Avramopoulos, I., Suchara, M., and Rexford, J. How small groups can secure interdomain routing. Tech. rep., Princeton University, Nov. 2007.

[3] Bates, T., Gerich, E., Joncheray, L., Jouanigot, J.-M., Karrenberg, D., Terpstra, M., and Yu, J. Representation of IP routing policies in a routing registry. RFC 1786 (Informational), Mar. 1995.

[4] Butler, K., Farley, T., McDaniel, P., and Rexford, J. A survey of BGP security issues and solutions. *Proc. IEEE 98*, 1 (Jan. 2010), 100–122.

[5] Dimitropoulos, X., Krioukov, D., Fomenkov, M., Huffaker, B., Hyun, Y., claffy, K., and Riley, G. As relationships: Inference and validation. *SIGCOMM CCR 37*, 1 (Jan. 2007), 29–40.

[6] Dyn Research. Pakistan hijacks YouTube, 2008.

[7] Gao, L., and Rexford, J. Stable Internet routing without global coordination. *ACM SIGMETRICS 28*, 1 (Jun. 2000), 307–317.

[8] Gersch, J., and Massey, D. ROVER: route origin verification using DNS. In *Proc. IEEE ICCCN* (Jul/Aug. 2013).

[9] Gersch, J., Massey, D., and Papadopoulos, C. Incremental deployment strategies for effective detection and prevention of BGP origin hijacks. In *Proc. IEEE ICDCS* (Jun. 2014).

[10] Gill, P., Schapira, M., and Goldberg, S. Let the market drive deployment: A strategy for transitioning to BGP security. In *Proc. ACM SIGCOMM* (Aug. 2011).

[11] Gill, P., Schapira, M., and Goldberg, S. A survey of interdomain routing policies. *SIGCOMM CCR 44*, 1 (Jan. 2014), 28–34.

[12] Goldberg, S. Why is it taking so long to secure Internet routing? *ACM Queue 12*, 8 (Oct. 2014), 327–338.

[13] Hiran, R., Carlsson, N., and Gill, P. Characterizing large-scale routing anomalies: A case study of the China telecom incident. In *Proc. PAM* (Mar. 2013).

[14] Hiran, R., Carlsson, N., and Shahmehri, N. Crowd-based detection of routing anomalies on the Internet. In *Proc. IEEE CNS* (Sep. 2014).

[15] Hiran, R., Carlsson, N., and Shahmehri, N. PrefiSec: A distributed alliance framework for collaborative BGP monitoring and prefix-based security. In *Proc. ACM CCS WISCS* (Nov. 2014).

[16] Karlin, J., Forrest, S., and Rexford, J. Pretty good BGP: Improving BGP by cautiously adopting routes. In *Proc. IEEE ICNP* (Nov. 2006).

[17] Kent, S. An infrastructure supporting secure Internet routing. In *Public Key Infrastructure*, vol. 4043. 2006, pp. 116–129.

[18] Kent, S., Lynn, C., and Seo, K. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications 18*, 4 (Apr. 2000), 582–592.

[19] Konte, M., Perdisci, R., and Feamster, N. ASwatch: An AS reputation system to expose bulletproof hosting ASes. In *ACM SIGCOMM* (Aug. 2015).

[20] Lad, M., , Pei, D., Wu, Y., Zhang, B., and Zhang, L. PHAS: A prefix hijack alert system. In *Proc. USENIX Security* (Jul/Aug. 2006).

[21] Lepinski, M., and Kent, S. An infrastructure to support secure Internet routing. RFC 6480 (Informational), Feb. 2012.

[22] Lychev, R., Goldberg, S., and Schapira, M. BGP security in partial deployment: Is the juice worth the squeeze? In *Proc. ACM SIGCOMM* (Aug. 2013).

[23] Madhyastha, H., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., and Venkataramani, A. iPlane: An information plane for distributed services. In *Proc. OSDI* (Nov. 2006).

[24] Suchara, M., Avramopoulos, I., and Rexford, J. Securing BGP incrementally. In *Proc. ACM CoNEXT* (Dec. 2007).

[25] Vervier, P.-A., Thonnard, O., and Dacier, M. Mind your blocks: On the stealthiness of malicious BGP hijacks. In *Proc. NDSS* (Feb. 2015).

[26] Wahlisch, M., Schmidt, R., Schmidt, T. C., Maennel, O., Uhlig, S., and Tyson, G. RiPKI: The tragic story of RPKI deployment in the Web ecosystem. In *Proc. ACM HotNets* (Nov. 2015).

[27] White, R. Securing BGP through secure origin BGP. *The Internet Protocol Journal 6*, 3 (Sep. 2003), 15–22.

[28] Zhang, Z., Zhang, Y., Hu, Y. C., Mao, Z. M., and Bush, R. iSPY: Detecting IP prefix hijacking on my own. *ACM CCR 38*, 4 (Aug. 2008), 327–338.

[29] Zhao, X., Pei, D., Wang, L., Massey, D., Mankin, A., Wu, S. F., and Zhang, L. An analysis of BGP multiple origin AS (MOAS) conflicts. In *Proc. IMW* (Nov. 2001).

[30] Zheng, C., Ji, L., Pei, D., Wang, J., and Francis, P. A light-weight distributed scheme for detecting IP prefix hijacks in real-time. In *Proc. ACM SIGCOMM* (Aug. 2007).

Measurement-Based Coalescing Control for 802.3az

Angelos Chatzipapas* and Vincenzo Mancuso+
*Universidad Carlos III de Madrid, Madrid, Spain, +IMDEA Networks Institute, Madrid, Spain
Email: {angelos.chatzipapas, vincenzo.mancuso}@imdea.org

***Abstract*—IEEE 802.3az standard (EEE), is the energy-aware alternative to legacy Ethernet. To save energy by extending the sojourn in the *Low Power Idle* state of EEE, packet coalescing has been proposed. While coalescing improves by far the energy efficiency of EEE, it is still far from achieving energy consumption proportional to traffic. Moreover, coalescing can introduce high delays. In this work, we use sensitivity analysis to evaluate the impact of coalescing timers and buffer sizes, and to shed light on the delay incurred by adopting coalescing schemes. Accordingly, we design and study measurement-based coalescing control solutions that tune the coalescing parameters on-the-fly, thus adapting the link to the instantaneous load and controlling the coalescing delay experienced by the packets. Our results show that, by relying on run-time delay measurements, simple and practical adaptive coalescing schemes outperform traditional static and dynamic coalescing. Notably, our schemes double the energy saving benefit of legacy EEE coalescing and allow to control the coalescing delay.**

***Index Terms*—IEEE 802.3az; Coalescing; Data Centers; Efficiency; Sensitivity; Simulation.**

I. Introduction

More than 20% of the energy consumption in data centers is due to the network operation, which establishes network as the second biggest energy consumer in data centers [1]. While high-speed Ethernet cards constantly absorb a considerable part of a server's consumption—e.g., 10 $Gbps$ cards consume $\sim$15 W [2]—recent studies have shown that network links are underutilized: $\sim$40% are "comatose" and another $\sim$40% of the links are loaded no more than 10% [3]. Hence, there is a clear need for introducing a network-wide energy saving mechanism.

To this goal, IEEE 802.3az [4], known as Energy Efficient Ethernet (EEE), introduces a Low Power Idle state (LPI) for unutilized links. However, in terms of energy saving, EEE underperforms even under low traffic conditions due to LPI transitioning delays [5], [6] and thereby more advanced solutions are needed. Packet coalescing [7], [8] has been proposed to enforce longer sojourns in LPI state, thus improving the energy proportionality of EEE. However, coalescing has a cost, i.e., additional queueing delay for packets.

Using sensitivity analysis, this paper discusses the properties of coalescing techniques for EEE gigabit links, and proposes the design of delay-controlled adaptive coalescing schemes that effectively trade off energy saving and delay guarantees. Specifically, the work (i) analytically studies the performance of gigabit EEE links with coalescing using real data traces that have been captured in an operational web hosting center, (ii) proposes *measurement-based coalescing control* algorithms (MBCC) that almost halve the energy consumption of EEE links with respect to legacy coalescing, while maintaining the coalescing delay bounded and (iii) shows that significant economy can be achieved in a typical data center ($\sim$\$1.7M/year).

Our goal is to design a new class of adaptive coalescing algorithms for EEE links, namely MBCC. To achieve this goal, we analytically build on top of the analysis we presented in [6], which accurately models the behavior of coalescing buffers in gigabit EEE links with static coalescing parameters.[1] Specifically, our prior work [6] accounts for the fact that energy saving features of gigabit EEE links are triggered by the traffic activity in both link directions simultaneously. Namely, gigabit EEE links exhibit a *bidirectional behavior*. On the one hand, the model of [6] allows to estimate both the potential energy saving and the coalescing delay, but, on the other hand, it does not show how to configure the coalescing parameters optimally. Here, we derive a sensitivity analysis of the coalescing delay and energy saving with respect to the coalescing timer duration and the coalescing buffer size, and use it to design measurement-based control schemes that outperform legacy coalescing schemes.

Our new analytical study reveals the importance of coalescing parameters in different scenarios, and unveils that by adjusting the sole coalescing timer duration, it is possible to tune the link performance to achieve near-optimal energy saving, while incurring controlled coalescing delay.

Exploiting our analytical findings, *we design a simple measurement-based delay-controlled distributed adaptive coalescing scheme* in which network cards at the edge of the link coordinate by running a simple distributed algorithm to sense the delay incurred by packets. Our proposal uses the sensed delay as control signal to trigger the dynamic adaptation of the coalescing timer in the direction identified through the analysis.

Notably, our study goes beyond existing results on dynamic/adaptive coalescing [6], [9], [10]. Indeed, the key and novel feature of MBCC proposal, which makes it different from the class of dynamic algorithms studied in [6], is that we explicitly account, through measurements, for the delay experienced by packets in the EEE link.

With our approach, adaptive coalescing can outperform static coalescing by a large factor. We validate the superiority of our MBCC schemes with respect to other existing solutions by

[1]We focus on gigabit links because they present the most challenging behavior for both modeling and implementation of coalescing strategies, as explained in Section II, and they are the most commonly deployed links in data centers. However, the algorithms presented in this paper can be used for the whole EEE link speed range.

ISBN 978-3-901882-83-8

using real traffic traces that we have captured in an operational web hosting center.

The rest of the paper is organized as follows. Section II describes the basic functionality of EEE, with and without coalescing, and explains the basic results available for the modeling of gigabit EEE links. Section III presents a sensitivity analysis of the parameters of EEE with coalescing. In Section IV we design MBCC. In Section V we benchmark our schemes and legacy ones. In Section VI we discuss related work. Finally Section VII concludes the paper.

II. Background

A. EEE Gigabit links

The goal of EEE is to achieve *energy proportionality*, i.e., that energy consumption be proportional to link load. EEE introduces (i) a low power state (namely Low Power Idle - LPI), in which the link does not serve traffic and consumes about 10% of the energy consumed by legacy Ethernet, (ii) an Active state (state A) which performs like legacy Ethernet serving the traffic, (iii) a Sleep state (state S), which is the transition of the link from state A to state LPI, and (iv) a Wake Up state (state W) which is the transition from state LPI to state A. In LPI, a "Refresh" message is sent every T_q time units, in order to check the condition of the link (e.g., connection, interference level, synchronization, etc.) and therefore to save time and resources when the link resumes its activity. For different Ethernet speed, e.g., 100 $Mbps$, 1 $Gbps$, and 10 $Gbps$, EEE has different specifications and transition mechanisms among states. Next, we describe the interesting and most deployed case of 1 $Gbps$ links where, unlike in the other cases, the traffic in both link directions has to be taken into consideration in order to switch between states.

In Fig. 1 we can see the specific EEE state transition graph for gigabit links, in which states L and C are introduced to differentiate pure idle and idle-with-coalesced-packets during LPI, as described later in Section II-B for the case of coalescing operation. The gigabit EEE link can start the transition to state LPI (state S) only when both link directions are inactive. If there is no arrival during an interval T_s (time to switch-off part of the electronics and go to sleep) the link successfully enters state LPI. In contrast, if there is an arrival during the transition in either of the two directions, the link switches back to state A in order to serve the packet. The link remains in state LPI as long as there is no packet arrival. As soon as a packet arrives, the link transits back to state A, which takes T_w time units, i.e., the time required to switch on all electronic parts (state W). For gigabit EEE links, the minimum values (which are typically implemented) for T_s, T_w and T_q are 182 μs, 16 μs and 20 ms, respectively. Note that energy-saving operations of gigabit EEE links are equally affected by arrivals in any link direction, so we refer to such a behavior as *bidirectional*.

B. EEE links with coalescing

Studies of EEE [5], [6] have shown that it is very inefficient and it does not provide any significant energy saving benefit for network loads that exceed a few percents (>5%). The main reasons for this behavior are that (i) the interarrival time between packets may prevent the link to enter state LPI (inter-packet spacing less than T_s) and (ii) packet arrivals do not allow long sojourns in state LPI, and thus most of the time is spent in transitioning. For instance, gigabit links spend 12 μs to serve a 1500-byte packet, against the 182 μs plus 16 μs required for the transition from state A to state LPI and back to state A passing through states S and W.

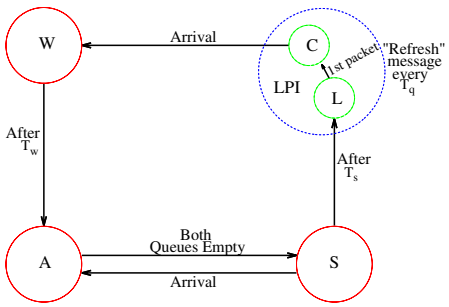


Fig. 1: State transitions for 1 $Gbps$ links with coalescing.

To face the above described issues, packet coalescing has been proposed. Coalescing prolongs the duration of state LPI since it introduces (i) two buffers of N_c packets, one for each link direction, where packets can be stored while the link is in state LPI and (ii) a timer of duration T_c which counts down from the arrival of the first packet in state LPI.

As depicted in Fig. 1, coalescing introduces two new states, which detail coalescing operations within state LPI: (i) state L, where the link enters after state S and in which it remains until it receives a packet in either of the two link directions, and (ii) state C, during which multiple packets are coalesced. Both in state L and state C, the link behaves (and absorbs low power) like in state LPI of a legacy EEE link.

The packet that triggered the transition from state L to state C starts the timer T_c, and the transition from state C to state W occurs after the timer T_c expires or when one of the two coalescing buffers fills up. We denote with τ_c the variable-size interval during which the link remains in state C.

C. Performance of Gigabit EEE links with coalescing

For the analysis presented in this paper, we build on the model presented in our prior work [6], which is the only accurate model that considers the bidirectional behavior of gigabit EEE links. For ease of presentation, here we report from [6] the expressions for the energy saving factor η_{LPI} and the average delay D_i for packets transmitted in direction i, where $i \in \{1,2\}$ indicates the two possible link directions:

$$\eta_{LPI} = \frac{\frac{1}{\lambda_1+\lambda_2} + E[\tau_c]}{E[T_{cycle}]}; \tag{1}$$

$$D_i = \frac{\sum_{\alpha\in\{A,S,L,C,W\}} n_\alpha^{(i)} D_\alpha^{(i)}}{\lambda_i E[T_{cycle}]}, \quad i \in \{1,2\}. \tag{2}$$

In the above expressions, the parameters λ_i represent the packet arrival rates in the two link directions, $E[\tau_c]$ is the average time that the link spends in state C, $E[T_{cycle}]$ is the average time spent between two consecutive transitions to state L (i.e.,

a system cycle), $n_\alpha^{(i)}$ corresponds to the number of packets received in link direction i in state α (denoting one of the states A, S, L, C, W), $D_\alpha^{(i)}$ is the average delay that the packets suffer in state α and direction i. From the results in [6], it is also possible to see that $\lambda_1 > \lambda_2 \Rightarrow D_2 > D_1$, so that the least loaded link suffers the highest delay. As concerns the duration of state C, we elaborate on the results of [6] and obtain the following expression for $E[\tau_c]$:

$$E[\tau_c] = \sum_{k=0}^{N_c-2} \sum_{j=0}^{N_c-2} \frac{\lambda_1^k \lambda_2^j}{k!j!} \int_{t=0}^{T_c} t^{k+j} e^{-(\lambda_1+\lambda_2)t} dt. \quad (3)$$

As it is clear from the above expression, $E[\tau_c]$ increases with both T_c and N_c, and so does $E[T_{cycle}]$, which strongly depends on $E[\tau_c]$. Below we report an approximation for $E[T_{cycle}]$ from [6], expressed as a function of loads and arrival rates in the two link directions, i.e., ρ_i and λ_i, respectively:

$$\begin{aligned} E[T_{cycle}] =& (T_w + E[\tau_c]) \left[1 + \frac{1}{\lambda_1+\lambda_2}\left(\frac{\lambda_1\rho_1}{1-\rho_1} + \frac{\lambda_2\rho_2}{1-\rho_2}\right)\right] \\ &+ \frac{e^{(\lambda_1+\lambda_2)T_s}}{\lambda_1+\lambda_2}\left[\frac{\rho_1}{1-\rho_1} + \frac{\rho_1^2(2-\rho_1)(\lambda_1\rho_2+\lambda_2)}{2\lambda_1(1-\rho_1\rho_2)(1-\rho_1)^2}\right. \\ &\left. + \frac{\rho_2}{1-\rho_2} + \frac{\rho_2^2(2-\rho_2)(\lambda_2\rho_1+\lambda_1)}{2\lambda_2(1-\rho_1\rho_2)(1-\rho_2)^2} + 1\right]. \quad (4) \end{aligned}$$

By defining two positive coefficients a and b, that only depend on arrival rates, loads, and EEE parameters T_w and T_s, the previous result can be expressed as a linear function:

$$E[T_{cycle}] = a + b\, E[\tau_c], \quad (5)$$

where a and b are constants that can be computed by comparing (4) and (5). In the above formulas, the dependency on T_c and N_c is concentrated in the term $E[\tau_c]$, therefore $E[T_{cycle}]$ grows with both T_c and N_c.

The analysis of $D_\alpha^{(i)}$ and $n_\alpha^{(i)}$ can be found in [6]. Here it is sufficient to recall that $D_L^{(i)}$, $D_C^{(i)}$, $D_W^{(i)}$, $n_C^{(i)}$ and $n_A^{(i)}$ can be expressed as a constant plus a term proportional to $E[\tau_c]$, and thus they also grow with T_c and N_c.

III. Sensitivity Analysis of EEE with Coalescing

We now proceed with a novel study on the sensitivity analysis of EEE performance with respect to the coalescing parameters. Specifically, we want to study the change of both energy saving and average packet delay when we modify either T_c or N_c. Thus, we apply the method of partial derivatives with respect to T_c and N_c.

The partial derivatives with respect to either T_c or N_c for both η_{LPI} and D_i show a dependence on the partial derivative of $E[\tau_c]$ as can be seen from the analysis presented in Section II-C. Thus, next we report the partial derivative of $E[\tau_c]$ (and $E[T_{cycle}]$), the rest is mere calculation.

A. Partial derivatives with respect to T_c

The partial derivative of $E[\tau_c]$ with respect to T_c is

$$\frac{\partial E[\tau_c]}{\partial T_c} = \sum_{k=0}^{N_c-2} \sum_{j=0}^{N_c-2} \frac{\lambda_1^k \lambda_2^j}{k!j!} T_c^{k+j} e^{-(\lambda_1+\lambda_2)T_c} > 0, \quad \forall T_c > 0; \quad (6)$$

and the partial derivative of $E[T_{cycle}]$ with respect to T_c is given by the following expression:

$$\begin{aligned} \frac{\partial E[T_{cycle}]}{\partial T_c} =& \frac{\partial}{\partial T_c}\left\{E[\tau_c]\left[1 + \frac{1}{\lambda_1+\lambda_2}\left(\frac{\lambda_1\rho_1}{1-\rho_1} + \frac{\lambda_2\rho_2}{1-\rho_2}\right)\right]\right\} \\ =& \left[1 + \frac{1}{\lambda_1+\lambda_2}\left(\frac{\lambda_1\rho_1}{1-\rho_1} + \frac{\lambda_2\rho_2}{1-\rho_2}\right)\right]\frac{\partial E[\tau_c]}{\partial T_c} \quad (7) \\ =& b\,\frac{\partial E[\tau_c]}{\partial T_c} > 0, \quad \forall T_c > 0. \quad (8) \end{aligned}$$

Finally, we get the partial derivative of the energy saving η_{LPI} with respect to T_c as follows:

$$\begin{aligned} \frac{\partial \eta_{LPI}}{\partial T_c} =& \frac{\frac{\partial E[\tau_c]}{\partial T_c} E[T_{cycle}] - \frac{\partial E[T_{cycle}]}{\partial T_c}\left(\frac{1}{\lambda_1+\lambda_2} + E[\tau_c]\right)}{E^2[T_{cycle}]} \\ =& \frac{a - \frac{b}{\lambda_1+\lambda_2}}{(a + b\,E[\tau_c])^2}\frac{\partial E[\tau_c]}{\partial T_c}. \quad (9) \end{aligned}$$

From the above expressions, it is clear that the energy saving is a monotonic function of T_c, and moreover $\frac{\partial \eta_{LPI}}{\partial T_c} > 0, \forall T_c > 0$. Therefore the delay monotonically increases with T_c.

Similarly, the partial derivative of the delay D_i with respect to T_c is:

$$\begin{aligned} \frac{\partial D_i}{\partial T_c} =& \frac{\left(\frac{\rho_i}{2\mu_i(1-\rho_i)} - D_i\right)\frac{\partial E[T_{cycle}]}{\partial T_c} - \frac{\rho_i}{2\mu_i(1-\rho_i)}\frac{\partial E[\tau_c]}{\partial T_c}}{E[T_{cycle}]} \\ &+ \frac{\left(\frac{1}{\lambda_1+\lambda_2} + T_w + E[\tau_c]\right)(1+\rho_i)\frac{\partial E[\tau_c]}{\partial T_c}}{E[T_{cycle}]}. \quad (10) \end{aligned}$$

Also in this case it is possible to show that $\frac{\partial D_i}{\partial T_c} > 0, \forall T_c > 0$ as far as loads are not extremely high.In practice, high loads prevent any EEE benefit [6], [7], [8], and therefore we can safely assume that the delay monotonically increases with T_c under the circumstances in which energy saving can be achieved.

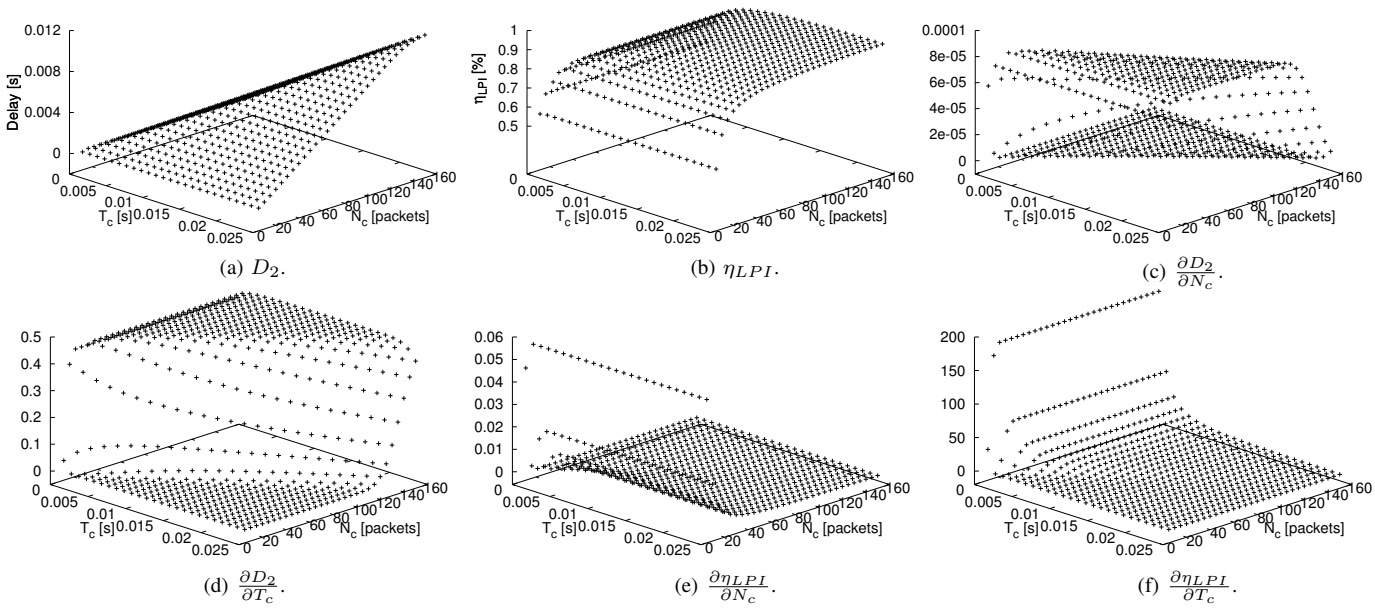
B. Partial derivative with respect to N_c

Regarding the partial derivative of $E[\tau_c]$ with respect to N_c, since N_c takes only integer values (it refers to packets) we consider the forward difference between $E[\tau_c]$ computed at $N_c + 1$ and at N_c:

$$\begin{aligned} \frac{\partial E[\tau_c]}{\partial N_c} \approx& \Delta_{N_c}[E[\tau_c]](N_c) = \frac{E[\tau_c](N_c+1) - E[\tau_c](N_c)}{1} \\ =& \sum_{j=0}^{N_c-2} g_{\lambda_1\lambda_2}(N_c-1, j) + \sum_{k=0}^{N_c-2} g_{\lambda_1\lambda_2}(k, N_c-1) \\ &+ g_{\lambda_1\lambda_2}(N_c-1, N_c-1) > 0, \quad \forall N_c \geq 2; \quad (11) \end{aligned}$$

where $g_{\lambda_1\lambda_2}(k,j) = \frac{\lambda_1^k\lambda_2^j}{k!j!}\int_{t=0}^{T_c} t^{k+j}e^{-(\lambda_1+\lambda_2)t} > 0, \ \forall T_c > 0$.

With the above, the partial derivatives of $E[T_{cycle}]$, η_{LPI}, and D_i with respect to N_c have the same form as their partial derivatives with respect to T_c. Therefore, we can conclude that energy saving and delay grow monotonically with N_c as well.



(a) D_2. (b) η_{LPI}. (c) $\frac{\partial D_2}{\partial N_c}$.

(d) $\frac{\partial D_2}{\partial T_c}$. (e) $\frac{\partial \eta_{LPI}}{\partial N_c}$. (f) $\frac{\partial \eta_{LPI}}{\partial T_c}$.

Fig. 2: Coalescing delay, energy saving, and their partial derivatives with respect to T_c and N_c. Since the delay due to coalescing is higher in the least loaded link direction, we show only the delay for packets transmitted in that direction (D_2).

TABLE I: Maximum η_{LPI} for $D_{target} \leq 1\ ms$ (ρ_1, ρ_2, λ_1, λ_2 are taken from real traffic traces)

ρ_1 [%]	ρ_2 [%]	λ_1 [pkts/s]	λ_2 [pkts/s]	$Max\{\eta'_{LPI}\}$ [%]	T'_c [ms]	N'_c [packets]	$Max\{\eta''_{LPI}\}$ [%]	T''_c [ms]	N''_c [packets]
0.11	5.25	2186	4343	82.09	≥ 3	≥ 32	82.09	=2	100
10.54	0.66	10410	5324	62.84	≥ 7	≥ 22	60.02	=2	100
0.57	32.68	10051	27459	15.34	≥ 9	$= 205$	8.74	≥ 5	100
1.01	40.52	17091	34042	1.80	≥ 10	$= 255$	0.75	≥ 4	100
5.06	0.5	5409	3809	77.50	≥ 5	≥ 15	66.55	$= 1$	100
1.14	17.93	9639	17320	37.59	≥ 7	≥ 75	31.17	$= 3$	100
0.20	0.06	310	268	92.72	$= 1$	≥ 15	92.72	$= 1$	100

C. Discussion

The partial derivatives with respect to either T_c or N_c show the strong dependency of D_i and η_{LPI} on $E[\tau_c]$ (and on $E[T_{cycle}]$ but this also depends on $E[\tau_c]$). Furthermore, the value of $E[\tau_c]$ grows with T_c and N_c, and we have shown that both η_{LPI} and D_i monotonically grow with T_c and N_c.

To graphically see the impact of T_c and N_c on the delay, D_i, and the energy saving, η_{LPI}, we plot in Fig. 2 an example of partial derivatives, representing the behavior of η_{LPI} and D_i for different T_c and N_c values when the offered load is $\rho_1 = 5.06\%$ and $\rho_2 = 0.5\%$. These loads correspond to a load profile of a traffic trace we collected on a gigabit link in a large web hosting center. Moreover, this is a representative link load since, according to [3], about 80% of the links operate with less than 10% of load, so that the selected case represents a medium load case.

Specifically, Fig. 2a illustrates the behavior of the delay experienced in the most loaded link direction (which is the highest of the two average delays). The figure shows that the delay quickly grows to unacceptable values with both T_c and N_c. The energy saving η_{LPI} also grows, but it does it faster with small values of T_c and N_c, and afterwards it saturates. Overall, the impact of T_c and N_c seems similar. However, the study of the partial derivatives presented in Fig. 2 unveils that both delay and energy saving are more sensitive to changes in T_c rather than in N_c. Indeed, Figs. 2c, 2d, 2e, and 2f point out that the partial derivatives with respect to T_c are up to three orders of magnitude higher than the ones with respect to N_c. We have observed the same behavior for a large range of load combinations, although the results are not shown here due to space limitations. Therefore, we can say that T_c is more important than N_c in the control of delay and energy saving in EEE. Another important observation is that the impact of N_c saturates for relatively small values of the coalescing buffer size, i.e., implementing buffer sizes of 100 packets allows to achieve the highest possible energy saving.

To validate the above observations, we report in Table I a few representative case studies corresponding to different combinations of average loads ρ_1 and ρ_2 as observed in real traffic traces for the two link directions. In the table, for each case, we report the maximum energy saving that can be achieved by manually varying T_c and N_c subject to an average delay below 1 ms (we denote with η'_{LPI} the energy saving factor that can be achieved subject to a given delay contraint). Additionally, we report the values T'_c and N'_c at which η'_{LPI} is maximized. Moreover, for the case without delay constraints but still the delay is below 1 ms, we fix the value of N_c to N''_c =100 packets (larger values do not improve the energy saving gain) and we check again the maximum value of the energy saving factor, which we denote as η''_{LPI}, achievable by varying T_c only. In the table, we report the value T''_c of the coalescing timer which maximizes the energy saving.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

From Table I, we can observe that energy saving in the two cases is very close, so that we can think of fixing the size of the coalescing buffer and using an adaptive coalescing algorithm that, by adjusting the sole coalescing timer T_c, is able to achieve near optimal energy savings while keeping bounded the average delay of the packets due to coalescing. Noticeably, Table I also shows that small values of T_c are needed to achieve optimal (or near-optimal) performance figures, so that the optimal value of T_c can be searched in a small range. We next use the results of this section to design a novel adaptive coalescing algorithm based on run-time delay measurements.

IV. Measurement-Based Coalescing Control

Differently from existing approaches, we use analytical results on the sensitivity of D_i and η_{LPI} to make *run-time* educated decisions on how to adapt the coalescing parameters to meet a maximum target delay D_{target}.

The analysis tells that T_c and N_c behave qualitatively in a similar way. Specifically, fixing one of the two parameters limits the maximum achievable energy saving, although, by tuning the other parameter, it is possible to adjust the energy saving from zero to the maximum while increasing the delay monotonically. Therefore, to implement an adaptive coalescing algorithm, it is enough to fix one parameter between T_c and N_c to a sufficiently high value (which guarantees that near-maximal energy saving can be achieved), and adapt the remaining parameter.

The analysis also unveils that η_{LPI} and D_i values are more sensitive to T_c rather than toN_c. With the above consideration, jointly to the fact that N_c is limited to integer values, T_c results to be a better candidate for the fine tuning of energy and delay tradeoff when coalescing is adopted.

Therefore, we design an adaptive coalescing algorithm in which only T_c is adjusted. Moreover, in our algorithm, we implement a simple yet effective mechanism to detect when the coalescing is causing excessive delay and timely react. What we include in the algorithm is a low-pass filter to estimate the average coalescing delay D_i. When the link switches to state W, the dynamic timer algorithm tunes $T_c \in [T_c^{\min}, T_c^{\max}]$ based on the experienced (measured) average delay and D_{target}. The pseudocode of our heuristic is reported in Algorithm 1.

The analysis says that increasing T_c increases both η_{LPI} and delay at any load, so when the average delay is below or above the target, the algorithm increments or decrements the T_c value, respectively. The advantages of our approach are twofold: (i) given that T_c is tuned after exiting state C, the delay adaptation procedure is almost immediate (a few milliseconds), which allows to instantly react to changes in packet delay; (ii) our adaptive algorithm adapts quickly to any changes in traffic load simply by estimating the packet delay. Load variations occur very often in the daily patterns and so our simple T_c adaptation mechanism can produce great benefit for EEE.

With the above, we have defined not one but an entire class of delay-controlled MBCC algorithms, which differ in the way the value of T_c is tuned. For example, additive or multiplicative increases and decreases can be used. In the following, we simply use either (i) an additive increase/decrease approach with fixed step δ or (ii) an additive increase/multiplicative decrease approach with fixed additive step δ and multiplicative decrease percentage γ. We only focus on those two schemes because, first, multiple increase schemes provide less fairness than additive increase schemes and second, multiple increase schemes wildly oscillate and are a source of instability, thus leading to poor performance [11], [12]. Note that, due to the high sensitivity of delay and energy saving with respect to variations of T_c, the possible values of δ and γ have to be small enough to cause small changes in the adaptive timer.

Algorithm 1: MBCC: Adaptive Coalescing Timer

```
Input: run-time average estimate of delays D_1 and D_2
while C → W do
    if (D_1 && D_2) ≤ D_target then
        if T_c < T_c^max then
            T_c = max{T_c + δ, T_c^max}
    else
        if T_c > T_c^min then
            T_c = max{T_c − δ, T_c^min} or
            T_c = max{(1 − γ)T_c, T_c^min}
```

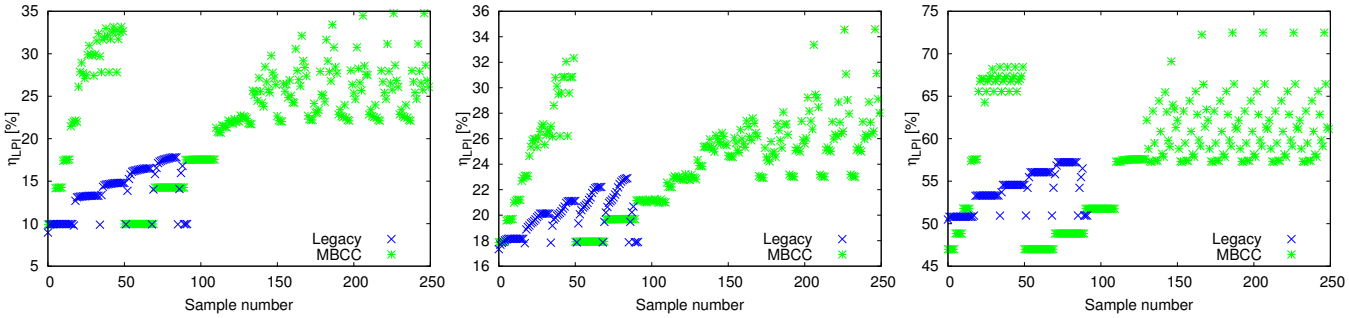
Note that, in gigabit EEE links, the two directions are correlated and therefore the algorithm has to run distributed over the two network cards at the edge of the link, although this requires only a few overhead messages to be transmitted from one card to the other to signal state transition events. However, such messages can be piggybacked by regular EEE state control messages, since each link edge just needs to send one bit to tell the other edge whether the measured delay is exceeding D_{target} or not.

V. Performance Evaluation

In this section we evaluate MBCC by implementing a delay-controlled adaptive coalescing timer algorithm. We benchmark MBCC against legacy static coalescing algorithms, for which it is known that dynamic adaptation based on coalescing events (such as buffer overflows or timer timeouts) does not improve performance if a target delay has to be guaranteed [6]. In the following, we first evaluate the achievable energy savings obtained by different configurations of legacy coalescing and MBCC for a set of representative traffic traces. Afterwards, we illustrate the behavior of energy savings and delays over time, when the load keeps changing. For our experiments, we use the real traffic traces we have been allowed to collect in Satec, a large web hosting center in Madrid, Spain.

A. Experimental setup

For our performance evaluation we monitored a typically low loaded link ($\rho_i \leq$10%) and a backbone link ($\rho_i \geq$10%) for a few minutes every one hour during a period of one year. Moreover, we modified the NS-3 simulator in order to (i) import the collected traces and (ii) simulate EEE with those traces as input, with both legacy coalescing and MBCC. Furthermore, we picked a few traffic traces, with loads



(a) $\rho_1 = 1.14\%$, $\rho_2 = 18\%$. (b) $\rho_1 = 0.57\%$, $\rho_2 = 32.7\%$. (c) $\rho_1 = 0.11\%$, $\rho_2 = 5.3\%$.

Fig. 3: Achievable energy saving for MBCC vs. Legacy coalescing with $D_{target} \leq 1\ ms$.

TABLE II: Legacy Coalescing: list of T_c and N_c combinations

Parameter	Value
T_c [μs]	200/500/700/1000/1200/**1300**/1400/1500/1700/2000
N_c [packets]	2/5/**10**/11/13/15/17/20/25/30/40/50/60/70/80/90/100

spanning from low to high, to further compare the achieved η_{LPI} using MBCC or legacy coalescing with bounded delay.

In our study, legacy coalescing schemes require the calibration of T_c and N_c based on the expected traffic characteristics or based on, e.g., the peak traffic. However, to guarantee low delay under low load conditions, both T_c and N_c have to be tuned to values well below the ones that achieve the best energy performance under medium or high traffic. In particular, since our criterion is to regulate the average coalescing delay of the packets crossing the EEE link under all traffic conditions, a (T_c, N_c) combination with small values has to be universally adopted to cope with the delay under scarce traffic conditions ($\sim$0.1% in the less loaded direction). Therefore, legacy coalescing has the disadvantage that it needs to be tuned on the *off-peak* traffic conditions. Apparently, so far, this has not been considered a great disadvantage for EEE links. In fact, energy savings are expected to be harvested only under low to medium traffic conditions. However, we argue that even though low loaded links represent about 40% of a data center links, there is still another 60% of the links from which additional energy savings could be potentially obtained.

To evaluate legacy coalescing, we test a range of values for T_c and N_c, as reported in Table II, under different traffic conditions. In contrast, for MBCC with our delay-controlled adaptive timer heuristic (Algorithm 1), we consider a fixed N_c value, such as the one selected based on the results reported in Table I (i.e., we could select the value N_c=100 packets), whereas the T_c value is automatically adapted according to the traffic. We assign D_{target} as initial value for the timer T_c. Other configuration parameters for MBCC are the adaptation coefficients δ and γ. The range of values for N_c, δ and γ can be read in Table III.

All tested parameters span over large intervals, to thoroughly explore their impact by means of our simulations.

B. Achievable energy saving

Here we compare legacy coalescing and MBCC under a variety of configuration choices, as reported in Tables II and III. In particular, we report our results for energy saving subject

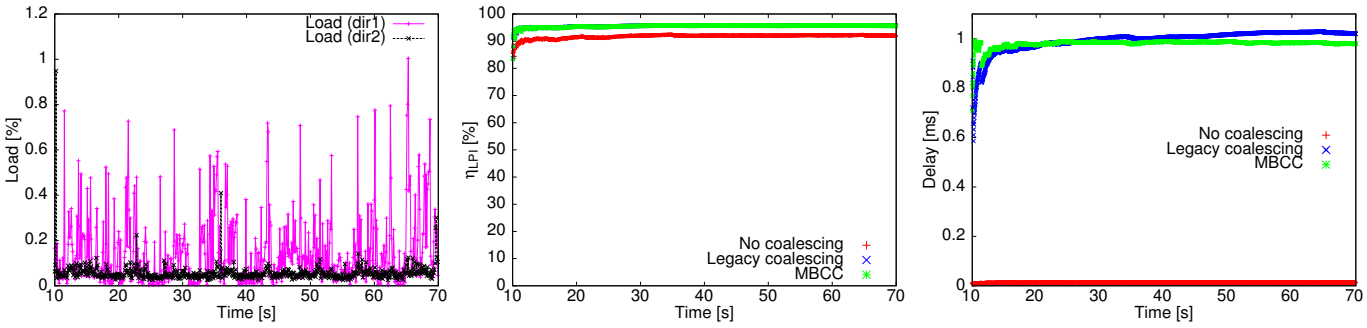
TABLE III: MBCC with Adaptive Coalescing Timer (Algorithm 1): list of parameters

Parameter	Value
δ [μs]	10/30/100/300/**1000**
γ [%]	**10**/25/50/75
N_c [packets]	2/5/10/20/50/75/100/200/500/**1000**

to average coalescing delay, D_{target}, not exceeding 1 ms. We think that this is a reasonable upper bound for the average delay in a point-to-point link. Indeed, according to [13] a connection between East and West coast in the US has at least four hops that create 28.4 ms of average delay. Thus, we consider that adding 1 ms due to the use of EEE in a data center connected to such a network is acceptable.

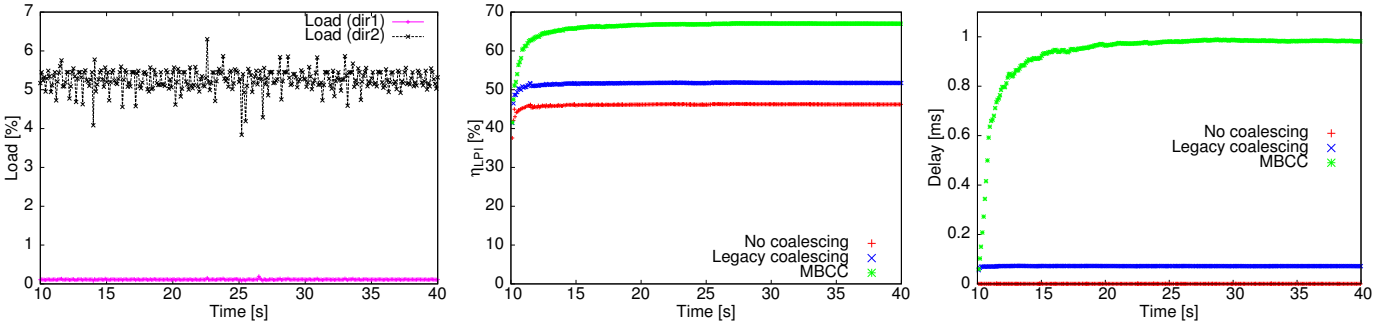
In Fig. 3 we plot η_{LPI} for three different load combinations $(\rho_1,\ \rho_2)$. For legacy coalescing we run the simulation of a trace with a given combination $(T_c,\ N_c)$ and we get the average delay of the packets in both directions and η_{LPI}. The delay can be higher or lower than D_{target} but we report the energy saving only for those combinations that give average coalescing delay below D_{target} (Table II). For MBCC, since it guarantees that the delay is below D_{target}, we report all points corresponding to all the tested combinations of parameters (Table III). Notably, the best results achieved with legacy coalescing in any of the depicted scenarios are very far from the best results of MBCC. Indeed, MBCC practically doubles the gain achieved by legacy coalescing.

Moreover, in our experiments we have observed that a particular combination performs best for legacy coalescing under any of the tested load combinations, i.e., (T_c=1300 μs, N_c =10 packets), reported in boldface in Table II. In contrast, for the case of MBCC, we have observed high variability in the configuration that achieves the best results in the various cases. In particular, considering that in each subfigure of Fig. 3 the first 50 samples for MBCC use only the parameter δ to adapt T_c (additive increase/decrease), while the remaining 200 samples use both δ and γ (additive increase, multiplicative decrease) for the adaptation of T_c, we can conclude that using both δ and γ is slightly more convenient. However, we have also observed that many configurations are equivalent, in particular when N_c is small (below 20), the performance is determined by N_c only, and changing δ and γ does not affect the results. In contrast, with higher N_c values δ and γ can be responsible for a fluctuation of 10-15% of energy saving. More in general,



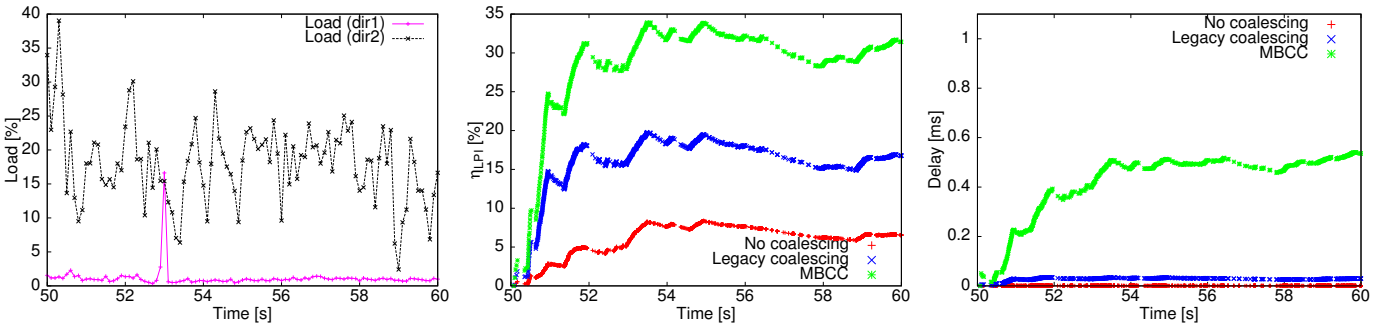
(a) Load. (b) Energy saving (Legacy and MBCC overlap). (c) Higher delay of the two directions.

Fig. 4: Low load ($\rho_1 = 0.2\%$, $\rho_2 = 0.06\%$). MBCC and legacy coalescing practically save the same amount of energy.



(a) Load. (b) Energy saving. (c) Higher delay of the two directions.

Fig. 5: Medium load ($\rho_1 = 0.11\%$, $\rho_2 = 5.3\%$). MBCC largely outperforms legacy coalescing at the expenses of delay (without exceeding the available delay budget).



(a) Load. (b) Energy saving. (c) Higher delay of the two directions.

Fig. 6: Highly variable load ($\rho_1 = 1.44\%$, $\rho_2 = 18.0\%$). MBCC doubles energy savings with respect to legacy coalescing.

our results indicate that bigger values of δ and N_c allow bigger energy saving. Instead, a bigger value of γ reduces the energy benefit. The topmost points in all the cases correspond to the combination ($N_c = 1000$ packets, $\delta = 1000$ μs, $\gamma = 10\%$), which is reported in boldface in Table III.

Now we select a near-optimal configuration for MBCC, and we compare its performance with the best configuration of the legacy coalescing scheme. We use an additive increase, additive decrease scheme with δ =100 μs, and N_c =100 packets for MBCC, and $T_c = 1300$ μs, and N_c =10 packets for legacy coalescing. With those configurations, in Figs. 4, 5, and 6 we plot the behavior over time of η_{LPI} and the higher of the two delays D_i for three different load combinations. In these figures, in addition to the performance MBCC and legacy coalescing, we also report the performance of EEE links without coalescing. Fig. 4 illustrates the case of low load. Specifically, as shown in Fig. 4a, the load in either link direction does not exceed 1%, and energy saving of 90-95% can be achieved with or without coalescing (see Fig. 4b). As concerns delay, Fig. 4c shows that coalescing introduces considerable delay with respect to the case of plain EEE without coalescing. However, the delay, D_{target}, is below 1 ms. The medium load case of Fig. 5 shows how MBCC manages to tradeoff delay for energy saving, while keeping the delay below 1 ms. Indeed, Fig. 5b shows the huge energy saving gain due to the delay-controlled coalescing operation of our proposal. In Fig. 6 we show a very dynamic case which combines high load with frequent and rapid load changes. We can still observe that our MBCC approach achieves a sevenfold gain with respect to plain EEE and a twofold gain with respect to legacy coalescing, while retaining the caused delay well below 1 ms. The impact of traffic variability is clear in the behavior of η_{LPI} and in the experienced delay.

Interestingly, the performance comparison shows that MBCC is able to maintain a constant gain over time with respect to the other schemes.

In conclusion, the energy benefit due to delay-aware MBCC is remarkable under any traffic condition, including under quickly variable traffic conditions. Configuring MBCC schemes is easy, since it only requires to make reasonably simple decisions on the maximum size of the coalescing buffer (in the order of 100 packets) and on the δ parameter (in the order of milliseconds). The γ parameter is optional and, if used, has to be chosen as a small factor (in the order of 10%).

C. Economical impact

The importance of EEE with MBCC can be seen in the following simple economical analysis. Let us consider a large data center, e.g., the one of OVH[2]. This data center contains 360,000 physical servers, and each server has on average 3 connected network ports [14]. Assuming that all network ports have gigabit links, each port may consume between 2 W and 13 W using legacy Ethernet [2]. Typical load distributions are $\sim$ 40% of the links at almost zero load ($\leq$ 0.1%), $\sim$ 40% between 0.1% and 10% of load, and the rest of the links operate at higher loads [3]. Therefore we can use the results of Figs. 4, 5 and 6 for an approximated economical analysis. Moreover, considering that the average cost of electricity in USA is about \$0.1/KWh, we can roughly estimate the cost of electricity for the network equipment of the servers of the aforementioned data center, using legacy Ethernet, plain EEE, EEE with legacy coalescing, or EEE with MBCC.

Thus, we will consider that on average an Ethernet card consumes 5 W and we further consider as averaged load values the ones we have in Figs. 4, 5 and 6. With our calculations, the annual electricity bill of data center servers just due to the network would be $\sim$\$4.73M using legacy Ethernet. This amount could be reduced almost by half accounting to $\sim$\$2.23M by adopting EEE. EEE with legacy coalescing could further deduct another $\sim$\$133K from the bill and, finally, MBCC could allow to save another $\sim$\$400K resulting in a final bill of $\sim$\$1.7M per year. Therefore, the adoption of MBCC could potentially reduce the electricity cost of a data center by $\sim$ 65% if compared with legacy Ethernet and by $\sim$ 25% if compared with plain EEE. Practically, MBCC would quadruplicate the cost saving attainable with legacy coalescing.

In this simple estimate we exclude switches and other equipment such as air conditioning, CPU processing or server fans which could further contribute to the electricity cost reduction. Moreover faster Ethernet cards, i.e., 10, 40 and 100 $Gbps$, consume even more energy (at least two, three and five times more, respectively), so that the potential for energy saving is greater for higher data rates.

The cost of implementing coalescing is just adding a buffer to the NIC to support the packet aggregation but this might not be a problem since NICs have already integrated memory buffers and thus all we need is to reserve some space for coalescing. The cost of measurement-based coalescing control is negligible since it only requires software modifications on the driver side in order to apply the timer adaptation. Therefore, we believe that EEE with MBCC adjusting the coalescing timer is worth further research interest.

[2]OVH.com presentation: http://www.youtube.com/watch?v=4e97g7_qSxA

VI. Related Work

Since the standardization of 802.3az in 2010 a few works appeared in the literature that try to model its behavior and predict accurately the amount of energy saving and the experienced delay both for EEE and EEE with coalescing.

*1) **EEE modeling**:* There are works which model EEE with a good accuracy, although they do not consider the effect of coalescing. Among them, [15], [16], [17] are the most representative. In [15] the authors present an $M/G/1$ model for 1 $Gbps$ links with unidirectional traffic. In [16] a two state model is proposed for unidirectional traffic and transition times are assumed to be multiples of the frame transmission time. Bolla et *al.* [17] present a complete framework for EEE links, from 100 $Mbps$ to 10 $Gbps$, that takes into account the bidirectional nature of 1 $Gbps$ links.

As already discussed, packet coalescing techniques promise the largest energy saving gain, and thus analytical models exist to predict their behavior. The first work that showed the outperformance of EEE with coalescing over the legacy EEE is [7]. The authors analyze the energy consumption improvement of the link using a buffer of 10 or 100 packets at the cost of limited additional delay. In [18] the authors develop a $GI/G/1$ model which approximates the energy saving and the delay that the packets suffer due to coalescing. The authors of [19] develop a $D/D/1$ model to estimate the energy consumed and the corresponding average delay of the packets, although they evaluate their model only with synthetic Poisson traffic. Kim et *al.* [20] present a similar mathematical analysis and evaluation based on synthetic traffic but using an $M/G/1$ queueing system. In [21] Meng et *al.* show a markovian model for 10 gigabit links and only big 1500-byte packets which estimates the energy saving and the average maximum delay of the first packet. In all the above mentioned models, the dependency of EEE operations on the traffic in both link directions is neglected, so that they cannot be realistically used for gigabit links with coalescing.

Differently from other proposals mentioned above, in [6] the authors propose a model specifically designed for gigabit links with coalescing. The model is based on the correlated behavior of two $M/G/1$ queues. Using simple parameters such as average packet size and average load, the model is able to estimate the energy consumption and the coalescing delay when coalescing parameters are static. The results of [6] also show that static coalescing offers performance levels as high as dynamic coalescing algorithms. However, that paper does not consider the class of measurement-based algorithms for the control of coalescing parameters that we propose in this paper. Indeed, here we have extended the analytical results of [6] to show the advantages of MBCC schemes for dynamic coalescing.

*2) **Dynamic Coalescing**:* This research area, i.e., EEE with dynamic coalescing is very new and quite active. Google has recently patented a series of adaptive algorithms in [9] for 10 gigabit links. In particular they suggest a modified version of EEE in which states A and LPI have fixed intervals and those intervals can only be adapted by a term Δ based on the type of data traffic to be transmitted. In [10] the authors propose a dynamic coalescing queue algorithm that adapts the buffer size N_c according to the difference between an ideal energy proportional saving model and the one proposed in their paper, but it lacks of complete performance evaluation using different parameters for the dynamic queue part. Moreover the results they provide show that static coalescing outperforms or at least achieves results similar to the dynamic scheme. The authors in [6] proposed two dynamic coalescing algorithms that adapt the coalescing timer T_c and the coalescing buffer size N_c, respectively. The event that triggers the adaptation of the corresponding parameter is either the timer expiration, or the fill-up of the buffer, with no further considerations on the network performance. For instance, T_c is always increased after a timer expiration, while N_c is always incremented if the coalescing buffer N_c fills up. That paper studies various parameters for timer and buffer size increase and decrease and, differently from our new work, [6] concludes that dynamic schemes do not outperform legacy coalescing. In [22] the authors propose an adaptive scheme to adapt the duration of the coalescing timer for passive optical networks which adopt EEE, based on a neural network-based algorithm which optimizes the duration of the state LPI versus the Wake-Up time. However, this scheme does not consider delay, which is the key factor for the applicability of EEE.

VII. Conclusion

In this paper we have used sensitivity analysis to understand the impact of coalescing parameters, such as timer T_c and buffer size N_c, on the energy saving and the delay experienced over Energy Efficient Ethernet (EEE) links with coalescing. The analysis reveals that optimizing energy saving subject to delay constraints is possible by simply adapting T_c. Therefore, based on the coalescing properties analytically studied, we have designed MBCC, a class of adaptive coalescing algorithms which adapts T_c according to the delay sensed by the link. MBCC achieves dramatic gain with respect to legacy coalescing algorithms, for which dynamic adaptation has been proven unnecessary and unfruitful. Specifically, we validated the superiority of MBCC with real traffic traces collected in a large web hosting center, and we showed that our proposal can even double the energy saving benefit with respect to legacy coalescing schemes. Moreover, from a purely economical point of view, MBCC can reduce the electricity cost of a data center by 65%. Notably, if compared to EEE with legacy coalescing, MBCC would quadruplicate cost savings on large data centers' electricity bill.

Acknowledgments

This work was supported by the Spanish Ministry of Economy and Competitiveness under the Ramon y Cajal Grant (ref: RYC-2014-01335), and under Grant TEC201455713-R (HyperAdapt).

References

[1] J. Arjona Aroca, A. Chatzipapas, A. Fernández Anta, and V. Mancuso, "A measurement-based analysis of the energy consumption of data center servers," in *ACM e-Energy '14*, Jun. 2014, pp. 63–74.

[2] R. Sohan, A. Rice, A. Moore, and K. Mansley, "Characterizing 10 Gbps network interface energy consumption," in *IEEE LCN 2010*, Oct. 2010.

[3] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, Jan. 2010.

[4] IEEE Std. 802.3az, "Energy Efficient Ethernet," 2010.

[5] P. Reviriego, K. Christensen, J. Rabanillo, and J. A. Maestro, "Initial evaluation of Energy Efficient Ethernet," *IEEE Communications Letters*, vol. 15, no. 5, pp. 578–580, May 2011.

[6] A. Chatzipapas and V. Mancuso, "Modelling and real-trace-based evaluation of static and dynamic coalescing for Energy Efficient Ethernet," in *ACM e-Energy '13*, May 2013, pp. 161–172.

[7] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro, "IEEE 802.3az: The road to Energy Efficient Ethernet," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 50–56, Nov. 2010.

[8] P. Reviriego, J. A. Maestro, J. A. Hernandez, and D. Larrabeiti, "Burst transmission for Energy Efficient Ethernet," *IEEE Computer Society*, vol. 14, no. 4, pp. 50–57, Jul. 2010.

[9] W.-C. Chang, W.-C. Lo, C.-S. Li, and M. Chang, "Adaptive pause time Energy Efficient Ethernet PHY," Jan. 2015, uS Patent 8,942,144.

[10] S. Herrería-Alonso, M. Rodríguez-Pérez, M. Fernandez-Veiga, and C. López-García, "Bounded energy consumption with dynamic packet coalescing," in *IEEE NOC 2012*, Jun. 2012, pp. 1–5.

[11] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM computer communication review*, vol. 18, no. 4, 1988, pp. 314–329.

[12] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN systems*, vol. 17, no. 1, pp. 1–14, 1989.

[13] B.-Y. Choi, S. Moon, Z.-L. Zhang, K. Papagiannaki, and C. Diot, "Analysis of point-to-point packet delay in an operational network," *Computer networks*, vol. 51, no. 13, pp. 3812–3827, Sep. 2007.

[14] S. Bapat, "The future of data centers (... and the stuff that goes in them)," in *1st Berkeley E3S Symposium*, Jun. 2009.

[15] M. Ajmone Marsan, A. Fernandez Anta, V. Mancuso, B. Rengarajan, P. Reviriego Vasallo, and G. Rizzo, "A simple analytical model for Energy Efficient Ethernet," *IEEE Communications Letters*, vol. 15, no. 7, pp. 773–775, Jun. 2011.

[16] D. Larrabeiti, P. Reviriego, J. A. Hernandez, J. A. Maestro, and M. Uruena, "Towards an energy efficient 10 Gb/s optical Ethernet: Performance analysis and viability," *Optical Switching and Networking*, vol. 8, no. 3, pp. 131–138, Mar. 2011.

[17] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, and P. Lago, "A closed-form model for the IEEE 802.3az network and power performance," *IEEE JSAC*, vol. 32, no. 1, pp. 16–27, Jan. 2014.

[18] S. Herrería-Alonso, M. Rodríguez-Pérez, M. Fernández-Veiga, and C. López-García, "A GI/G/1 model for 10Gb/s Energy Efficient Ethernet links," *IEEE Transactions on Communications*, vol. 60, no. 11, pp. 3386–3395, Nov. 2012.

[19] M. Mostowfi and K. Christensen, "An energy-delay model for a packet coalescer," in *IEEE Southeastcon*, Mar. 2012.

[20] K. J. Kim, S. Jin, N. Tian, and B. D. Choi, "Mathematical analysis of burst transmission scheme for IEEE 802.3az Energy Efficient Ethernet," *Elsevier Performance Evaluation*, vol. 70, no. 5, pp. 350–363, May 2013.

[21] J. Meng, F. Ren, W. Jiang, and C. Lin, "Modeling and understanding burst transmission algorithms for Energy Efficient Ethernet," in *IEEE/ACM IWQoS 2013*. IEEE, 2013, pp. 1–10.

[22] S. Lee and K.-Y. Li, "Adaptive state transition control for energy-efficient gigabit-capable passive optical networks," *Photonic Network Communications*, Apr. 2015.

Consolidating Flows with Implicit Deadlines for Energy-Proportional Data Center Networks

Xiaoda Zhang*, Zhuzhong Qian*, Sheng Zhang*, Kui Wu†, Sanglu Lu*
*State Key Laboratory for Novel Software Technology, Nanjing University, China
†Department of Computer Science, University of Victoria, Canada
Email: zhangxiaoda@dislab.nju.edu.cn, {qzz, sheng, sanglu}@nju.edu.cn, wkui@cs.uvic.ca

***Abstract*—To reduce the energy consumption of a large number of network devices in a data center, energy-efficient schemes use various heuristics to consolidate traffic to fewer switches. Most of these works, however, ignore the flow-level performance, which is one of the most critical requirements in production data centers. Hence, flow rate allocation should be considered together with flow path selection to guarantee flow-level performance and in the meantime save energy of network devices. For this reason, we present a framework to ensure that the energy consumption for data center network (DCN) is proportional to the traffic and to guarantee the flow-level performance. Our solution consists of two components: (i) flow rate allocation to meet flows' deadlines and (ii) flow path selection to use fewer switches. We compare our framework with existing techniques under synthetic traffic patterns. Results show that our framework could save, on average, 20% of network energy than the always-on baseline, while maintaining the better flow-level performance, and achieving good running time and fault tolerance simultaneously.**

I. Introduction

High energy consumption has become a central issue for large-scale data centers as computing and networking infrastructures scale out in response to growing requests in clouds. It is shown in [1] that the energy used in U.S. data centers in 2013 was estimated 91 billion kwh and is projected to increase to roughly 140 billion kwh annually by 2020. Various techniques, including Dynamic Voltage Frequency Scaling (DVFS), virtualization and efficient power supplies, have been explored to reduce the energy consumed by servers which accounts for the largest component of a data center's total power. The network devices, which could account for 10-20% of a data center's total power [9], [13], also need energy saving schemes.

DCNs are usually provisioned for the peak workload, and this load exceeds their long-term utilizations by a large margin. Therefore, significant energy could be saved if the energy consumption of a DCN could be proportional to its actual rather than peak workload. We call a DCN with this feature an *energy propoertional* DCN. A power measurement [16] studied several data center switches under a variety of traffic patterns. The study showed that keeping a switch always on consumes most energy, while increasing the traffic from zero to full load via a switch only increases the switch's power by less than 10%. This phenomenon implies that to achieve DCN energy proportionality, we could mainly focus on the number of power-on switches instead of the traffic load going through the switches. DCN topology designs, *e.g.*, Fat-tree [2], VL2 [10], BCube [11] provide more network components and more paths between arbitrary pairwise servers. This advantage brings opportunities to improve DCNs energy-proportionality, because turning off a subset of switches would not disconnect servers.

ISBN 978-3-901882-83-8 © 2016 IFIP

While traffic consolidation has been an effective way to achieve energy-efficiency by consolidating flows to fewer switches [13], [19], [21], they heavily depend on the accurate prediction of traffic [7]. On the other hand, most of the traffic engineering based solutions are agnostic to the network flow performance, which results in delaying flows and slowing down responses to requests. This degradation is not acceptable for applications requiring high quality of services (QoS).

An important class of data center applications, called Online Data-Intensive (OLDI) applications, *e.g.*, web search and online retail, employ algorithms where every query operates on data spanning thousands of servers. Latencies in this request-response process would heavily affect users experiences. To avoid the performance degeneration and keeping DCNs energy efficient, we avoid monitoring traffic flows frequently but instead obtain the flows' deadlines implicitly with the TCP AIMD mechanism. Quantitatively, we measure the flow performance in terms of its flow completion time (FCT), as the previous works did [14], [17], [18], [22]. Our objective is to design state changing (on/off) schemes for switches where the DCN energy consumption is minimized and the flow deadlines are met.

In order to reduce a DCN energy consumption, the optimal solution should meet the following goals:

- *Work Conservation* (*WC*): the principle to keep active switches with high utilization.
- *Performance Guarantee* (*PG*): the criterion for energy saving schemes that should not affect network FCTs.
- *Network Agility* (*NA*): the ability to dynamically grow and shrink the DCN capacity to meet traffic loads.

In this paper, we formulate this DCN Energy Saving (DES) problem and prove its NP-completeness. To address the DES problem, we propose a framework to optimize the energy consumptions while maintaining the network performance. To find the most suitable network subset, we present multiple algorithms to select paths for flows such that the flow bandwidth demands are satisfied and the number of required switches

is minimized. Each algorithm achieves different tradeoffs between efficiency and optimality. To keep the flow performance, the transmission rates are allocated in a way which meets the flow deadlines and efficiently utilizes the bandwidths. Using flow rate allocation and flow path selection, our framework could dynamically adjust the active set of network elements to satisfy changing traffic loads. Experiments show that compared to existing works, our solution on average, could save 20% of the network energy in data centers, while holding the better flow performance.

The rest of the paper is organized as follows: Section II formulates the DES problem and analyzes its hardness. Section III describes the framework where the rate allocation and path selection algorithms are proposed. Section IV presents the simulation results. We review the related works in Section V and conclude the paper in Section VI.

II. The DCN Energy Saving Problem

In this section, we first analyze the characteristics of DCN switch and the power-down strategy. Based on the preliminaries, we formulate the DES problem as a constrained optimization problem, and analyze its complexity.

A. Preliminaries

A network switch is commonly composed with chassis, line-cards, switching fabric, and ports. The chassis usually consumes a constant power. The line-card buffers packets and the switching fabric maintains the switching table. Ports consume dynamic power according to their speeds. The DES problem critically relies on the nature of the speed-power curve $f(s)$, a function mapping the switch's processing speed s to its power. In [4], [5], [20], the authors calculate $f(s)$ only by switch ports. In this work, $f(s)$ is more general and includes a constant plus a dynamic speed-related component. This model meets the real statistics [16] and supports our idea that, powering down the unused switches can save more energy than speed scaling [4]. Equivalently, we transfer the port power consumption to its associated link power consumption.

Advanced architectures (*e.g.*, Fat-tree [2], VL2 [10], BCube [11]) enrich the connectivities among servers in data centers. But the measurements from [7] show that, the utilization of aggregation switch links is 8% at 90% running time, while the average utilizations of edge layer switches and core layer switches are around 20% and 40%, respectively. This enables us to combine flows from several low utilization switches. As Fig. 1 illustrates, flows could be consolidated onto fewer switches when the DCN is at low utilization. In this example, after flow consolidation, six idle switches could be turned off, reducing network power by nearly 30%.

B. Problem Formulation

We assume that flows K_1, K_2, ..., K_n are transmitted among the DCN. We denote the i-th flow, $K_i = \{s_i, t_i, d_i\}$, where s_i is the source, t_i is the sink, and d_i is the flow size. The DCN is abstracted as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, including hosts and switches, and $\mathcal{E}$ is the set of

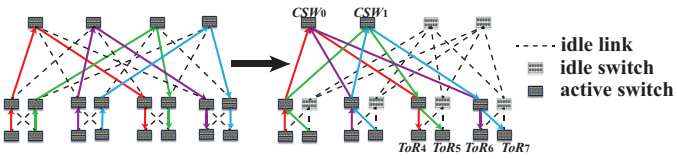


Fig. 1. An example of consolidating flows onto fewer switches in a Fat-tree topology using 4-port switch

links connecting them. Each link $(u,v) \in \mathcal{E}$ has the capacity $c(u,v)$. We do not allow flows to get split due to the fact that the reordering packets would degrade TCP performance, as previous works did [3], [13]. $f_i(u,v)$ is the size of the i-th flow along link (u,v), which is either d_i or 0. An assignment of flows is a mapping from flows to paths in the DCN, such that the following constraints are satisfied.

$$\forall (u,v) \in \mathcal{E}, \sum_{i=1}^{n} f_i(u,v) \leq c(u,v) \tag{1}$$

$$\forall i, \sum_{r \in \mathcal{V}} f_i(u,r) = 0, \text{when } u \neq s_i \text{ and } u \neq t_i \tag{2}$$

$$\forall i, \sum_{w \in \mathcal{V}} f_i(s_i,w) = \sum_{w \in \mathcal{V}} f_i(w,t_i) = d_i \tag{3}$$

The constraint (1) restricts the total flows along each link not exceeding the capacity. The flow conservation constraint expressed in (2) means that flows should not be created or destroyed at intermediate nodes. Equality (3) means that the sink receives the same amount of data as that the source sends.

Next, we define the following notations:

- $\mathcal{S}$: the set of all switches;
- P_u: the power of switch u, including the *constant* component plus the *dynamic* component;
- P_u^{cons}: the *constant* power component of switch u;
- $P_{u,v}^{link}(x_{u,v})$: the *dynamic* power component consumed by link (u,v), which is related to the flow rate $x_{u,v}$ on it;
- X_u: the indicator deciding whether the switch u is on;
- $Y_{u,v}$: the indicator deciding whether the link (u,v) is on;
- $R_i(u,v)$: the indicator deciding whether the i-th flow uses link (u,v).

The objective function can be formulated as follows:

$$\min \sum_{u \in \mathcal{S}} X_u \cdot P_u, \tag{4}$$

where

$$P_u = P_u^{cons} + \frac{1}{2} \sum_{v \in \mathcal{S}_u} Y_{u,v} \cdot P_{u,v}^{link}(x_{u,v}), \tag{5}$$

$$x_{u,v} = \sum_{i=1}^{n} f_i(u,v). \tag{6}$$

A factor of $\frac{1}{2}$ in Equation (5) is to eliminate the double counting of each link. For the link power consumption model

$P_{u,v}^{link}(x_{u,v})$, we adopt the power function from [4], a widely used version in the literature:

$$P_{u,v}^{link}(x_{u,v}) = \sigma + \mu x_{u,v}^{\alpha}, \tag{7}$$

where σ, μ and α are constants, and usually $\alpha \geq 1$. To avoid the stability problem incurred by frequently togging on and off links, we assume that a link can be powered off only when it carries no traffic, thus the constraint (1) can be augmented with binary variable $Y_{u,v}$:

$$\forall i, \forall (u,v) \in \mathcal{E}, \sum_{i=1}^{n} f_i(u,v) \leq Y_{u,v} \cdot c(u,v). \tag{8}$$

Actually, when all links from a switch are powered off, the switch can be powered off too:

$$\forall u \in \mathcal{S}, X_u = 1 - \prod_{v \in \mathcal{S}_u} (1 - Y_{u,v}). \tag{9}$$

Since we do not allow flow splitting, we have:

$$\forall i, \forall (u,v) \in \mathcal{E}, f_i(u,v) = R_i(u,v) \cdot d_i. \tag{10}$$

Data center network Energy Saving (DES) Problem: Given flows K_1, K_2, ..., K_n, decide X_u ($\forall u \in \mathcal{S}$), where (4) is minimized and constrains (1)(3)(8)(9)(10) are satisfied.

C. The Hardness of DES Problem

Theorem 1: The DES problem is NP-complete.

Proof: First we consider the adapted-DES problem, where we assume the constant component of the power of switch P_u^{cons} could be evenly partitioned to the active links, and the power of link $P_{u,v}^{link}(x_{u,v}) = \sigma + x_{u,v}$, with $\mu = 1$, $\alpha = 1$, and $\sigma = P_u^{cons}/|\mathcal{S}_u|$. Hence, the objective function of this adapted-DES is:

$$\sum_{(u,v) \in \mathcal{E}} (\sigma + x_{u,v}),$$

which is equal to

$$\sum_{(u,v) \in \mathcal{E}} x_{u,v} + \sum_{u \in \mathcal{S}} P_u^{cons}.$$

The Multi-Commodity Flow (MCF) problem is to minimize

$$\sum_{(u,v) \in \mathcal{E}} a(u,v) \cdot x_{u,v},$$

while satisfying constraints (1)(2)(3). Any MCF problem instance could be reduced to the adapted-DES instance in polynomial time, by letting $a(u,v) = 1$ and adding an additional constant. As the MCF problem is NP-complete for integer flows [8], the adapted-DES problem is NP-complete. Any additional constraints like P_u^{cons} cannot be shared by links or $\alpha > 1$, making the DES problem not easier than the adapted-DES problem. ∎

Due to the hardness of DES problem, we explore practical and efficient schemes to improve DCN energy-efficiency and flow performance.

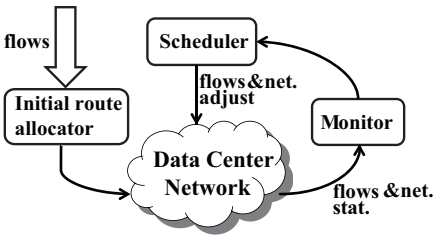


Fig. 2. System framework overview

III. Energy Saving Schemes for DCNs

First we introduce our system framework, as shown in Fig. 2. Typically, flows are initialized by initial route allocator. As flows arrive and depart, the monitor periodically detects the DCN configurations, and when the network gets suboptimal, the scheduler adjusts flow routes and the network configurations. In this adjusting process, we decouple rate[1] allocation and path selection for flows, aiming at satisfying flow rate demands and minimizing the occupied network simultaneously. For rate allocation, we first apply a simple method to label *implicit* deadlines to deadline-agnostic flows, and use them to calculate the bandwidth demands (III.*A*). Based on the bandwidth demands, linear programming and simulated annealing methods are proposed to select flow paths, and to output the minimum subset of the DCN (III.*B*). The actual rate that a flow obtains is the minimum allocated bandwidth along the selected path. We also extend the methods for practical considerations (III.*C*).

A. Flow Rate Allocation

1) Bandwidth demand calculation: One of the main metrics of the flow performance is its duration time, in other words, the flow completion time. For applications in data centers, especially OLDI applications, flows need to meet their *deadlines* to be useful. Nevertheless, deadlines are hard to explicitly acquire from the packet headers. Since the *sizes* of flows are easy to obtain, we can exploit the sizes of flows to estimate the implicit deadlines for flows, and then use them in bandwidth allocations. The method of calculating deadlines is based on the TCP AIMD mechanism.

When the network is lightly loaded, the TCP (Reno version) congestion window shows the sawtooth wave shape (Fig. 3). The area between the wave and the x-axis can be considered as the amount of data a flow sends. In a sawtooth wave (the gray region), the amount of data is around $\frac{3}{4}WL$ ($L \cdot \frac{W}{2} + \frac{1}{2} \cdot L \cdot \frac{W}{2}$). We assume that the window size starts from $\frac{W}{2}$, and increases linearly until congestion occurs. Given the flow size *A*, it is easy to approximate the flow duration time by a simple formula:

$$D = \frac{A}{\frac{3}{4}WL} \cdot L = \frac{4A}{3W}.$$

We regard *D* as the implicit deadline, and this value would be tighter than the time it really takes when transmitting in DCNs, due to the assumption on light network traffic. Given

[1]We use the term rate and bandwidth interchangeably.

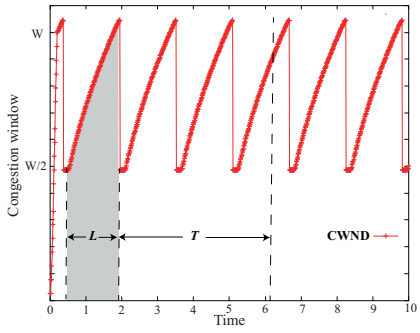


Fig. 3. TCP congestion window

a flow with remaining size $f_i.a$ and the remaining time until its deadline expires $f_i.r$, the demand bandwidth is estimated as $f_i.de = \frac{f_i.a}{f_i.r}$.

2) Rate allocation: The bandwidth demand request travels along the selected path to its destination, along which the nodes allocate bandwidths accordingly. The actual bandwidth flow f_i gets is the minimum of the allocated bandwidths. We apply the Greedy Bandwidth Allocation (GBA, Algorithm 1) for each node, which receives bandwidth requests from flows. If the node has sufficient capacity, each flow will acquire bandwidth $f_i.al = f_i.de + b$, where b is the spare bandwidth capacity shared with other flows. When the node does not have enough capacity to satisfy all the requests, GBA will try to satisfy requests as many as possible. The way we calculate $f_i.al$ respects *PG* goal and outperforms original TCP in terms of FCTs, as shown in Section IV.

Algorithm 1 Greedy Bandwidth Allocation

Input: $f[1...N]$: flows to be allocated, **B**: bandwidth capacity
Output: flows $f[1...N]$ with bandwidth allocations
1: Sort $f[1...N]$ by $f.de$ in a non-decreasing order;
2: i = 1, R = **B**;
3: **while** $i \leq N$ and $R > 0$ **do**
4: $f_i.al$ = **min**$\{R, f_i.de\}$;
5: $R = R - f_i.al$;
6: **end while**
7: **if** $R > 0$ **then**
8: $b = R/N$, i = 1;
9: **while** $i \leq N$ **do**
10: $f_i.al = f_i.al + b$;
11: **end while**
12: **end if**
13: Return $f[1...N]$;

B. Flow Path Selection

Based on the flow bandwidth demands, we have developed two methods to compute a minimum-power network subset in DCNs. The first is based on linear programming, with a randomized rounding scheme for minimizing active switches. Inspired by [3], the second method uses simulated annealing technique to get the solution by efficiently searching solution space. Each method includes both initial route allocator and scheduler algorithms.

1) Linear programming method: First we introduce our initial route allocator in this method. Energy-Efficient Routing (EER, Algorithm 2) counts the number of active switches in each path and sorts the paths in a non-increasing order. Next, EER checks the remaining bandwidths of the ordered paths ($p_i.re$) successively and allocates them to the candidate flow until satisfying its demand (*f.de*). The divisor (Λ − N[i]) (line 13) is small when there are many active switches in this path, and in this case the probability of choosing this path is high . We set Λ = 6 in Fat-tree, since there are 5 switches between inter-pod hosts. For example, $f.de$ = 20, N[1] = 5, N[2] = 2, A[1] = 10 and A[2] = 10, after *Normalize*(), the probability of choosing p_1 and the probability of choosing p_2 are 0.8 and 0.2, respectively. EER tends to allocate flows to paths which have more active switches and to meet *WC* goal.

Algorithm 2 Energy-Efficient Routing

Input: f: flow to be allocated, p: the set of M possible paths
Output: an ideal path chosen for f
1: **for** i: 1 to M **do**
2: N[i] = *CountActiveSW*(p_i);
3: **end for**
4: Sort p by number of active switches in decreasing order;
5: **for** i: 1 to M **do**
6: **if** $p_i.re \leq 0$ or $f.de \leq 0$ **then**
7: continue;
8: **end if**
9: N[i] = *CountActiveSW*(p_i);
10: A[i] = **min**$\{p_i.re, f.de\}$;
11: $p_i.re = p_i.re$ − A[i];
12: $f.de = f.de$ − A[i];
13: A[i] = A[i] / (Λ − N[i]);
14: **end for**
15: *Normalize*(A[1...M]);
16: Return p_i with probability A[i];

As flows arrive and depart, the network utilization may become suboptimal. The scheduler takes the network configuration and the flows as inputs and recomputes the paths for flows in nearly real-time, such that the active switches are minimized and the *NA* goal is respected. For scheduler algorithm in this method (LP, Algorithm 3), LP solves the MCF problem with fractional linear programming, and then extracts candidate paths for each flow. With the fractional results, we use a natural randomized rounding scheme, where the paths are chosen with the probabilities depending on their weights. Similarly to EER, we measure the weight of the j-th candidate path for the i-th flow by the assigned bandwidths $\mathcal{P}_i[j].as$ and the number of active switches N[j], giving that LP respects *WC*.

2) Simulated annealing method: Directly computing the flow assignment needs exhaustive search in the solution space, which is exponential to the number of flows. We introduce

Algorithm 3 Linear Programming Based Path Selector

Input: $f[1...N]$: flows to be allocated, $\mathcal{G}$ = ($\mathcal{E}$,$\mathcal{V}$): the DCN
Output: routing path p_i for $f[i]$

Solve MCF problem by linear programming;
For each flow $f[i]$, extract a set of candidate paths $\mathcal{P}_i$;
for i : 1 to N **do**
  for $\mathcal{P}_i[j] \in \mathcal{P}_i$ **do**
    N[j] = *CountActiveSW*($\mathcal{P}_i[j]$);
    $w[i][j]$ = $\mathcal{P}_i[j].as$ / (Λ−N[j]);
  end for
  Normalize($w[i]$);
  Choose p_i with probability $w[i]$;
end for
Return p_1, p_2, ..., p_N;

a novel method which can significantly reduce the solution space. The key insight is that a core switch can handle multiple flows destined to specific hosts. In Fig. 1, CSW_0 handles flows destined to hosts under ToR_4 and ToR_6, while CSW_1 handles flows destined to hosts under ToR_5 and ToR_7. That is, we shift from choosing paths for flows to choosing core switches for hosts. Once the mapping from hosts to core switches is determined, so are the paths for flows. Both initial route allocator and scheduler algorithm in this method depend on this mapping, since initial route allocator could initialize paths for flows by simply searching in this mapping in terms of their destinations, and scheduler updates this mapping by calling SA (Algorithm 4).

SA is to find the optimal state in the state space, where a state is a particular mapping. In each iteration, we generate a neighboring solution from current state and accept it as the new state with a probability depending on the energy of current and neighboring state, and current temperature. SA proceeds the iterations, with a decreased temperature each round, and it stops when the temperature reaches 0.

Algorithm 4 Simulated Annealing Based Path Selector

Input: s: initial state, n: iteration count
Output: s_B: beat state of the mapping

e = *Energy*(s);
s_B = s, e_B = e;
for T: n to 0 **do**
  s_N = *Neighbor*(s);
  e_N = *Energy*(s);
  if $e_N < e_B$ **then**
    s_B = s_N, e_B = e_N;
  end if
  if *Pr*(e_N−e, T) > *Rand*() **then**
    s = s_N, e = e_N;
  end if
end for

With the flow assignment determined by SA, we could easily find whether the flow demands in a link exceed its capacity. The *Energy*() function is defined by the total exceeded bandwidth demands and the number of active switches in s:

$$\mathit{Energy}(s) = (s.ex\_band + 1) \cdot (s.active\_sw + 1).$$

An extra 1 is necessary to ensure that the energy is always positive and comparable. *Pr*() defines the probability of accepting neighboring state as the new state:

$$\mathit{Pr}(\Delta E, T) = \begin{cases} 1 & \Delta E > 0 \\ e^{c\Delta E/T} & \Delta E \leq 0 \end{cases}$$

where c is a adjustable parameter.

Initial state: Particularly, the number of core switches is equal to the number of hosts in a pod in the Fat-tree. We restrict our first initial state to one-to-one mapping, which implies each host in a pod is mapped to a unique core switch. In subsequent scheduling phases, we set the initial state as the best state from the previous phase. This configuration could reduce the disruption of existing flows.

Neighbor state generator: Our neighbor state generator directs SA to appropriate mappings which saves more energy and tries to meet the capacity constraints. Our strategies are: (i) powering off a randomly-chosen core switch and remapping the involved hosts to other randomly-chosen core switches, (ii) swapping the mapping relationships of two randomly-chosen hosts, (iii) remapping a randomly-chosen host to another core switch, and (iv) powering on a core switch and remapping a randomly-chosen host from other switches to this switch. The first strategy is for energy-saving, while the last three strategies are for fine placements of flows with less exceeded capacity bandwidths. These four strategies are chosen with equal probability.

Energy function: The energy function we define involves both energy and bandwidth allocation information. Less energy indicates either less energy consumptions or less exceeded bandwidths, either of which would be accepted as a *good* state. Given a fixed traffic, less energy consumption mean less active switches, and then higher switch utilizations, giving that SA respects *WC*.

C. Practical Considerations

Stability consideration: The algorithms we have proposed are to find the minimum subset of switches, which achieve the three goals presented in Section I. However, these may go too far and result in unstable load. Unstable load will lead to different subsets of switches in the contiguous scheduling phases, meaning that the states of several switches change frequently. For stability consideration, we extend our algorithms with a *Hit mechanism*.

After applying the scheduler algorithm (LP or SA), we extract the path for each flow, with which the subset of switches is determined to run until next schedule phase. For a switch not in the subset, (*i.e.*, the switch would be turned off), if the number of requests for powering off this switch in the scheduling interval is greater than a given *threshold*, we actually power off this switch, otherwise, we keep the switch

on and randomly choose flows from geographically neighbor switches to adjust their paths to go through this switch. We correspondingly call the algorithms with *Hit mechanism* as LP-hit and SA-hit.

Fault tolerance consideration: The framework could certainly minimize the network energy consumption, but may hurt the performance regarding fault tolerance of the original system that always keeps switches on. We refer to the original system as *always-on baseline* in our later evaluation. In current data centers, failures are quite common due to hardware, software, and power outage problems [6], [12]. We thus improve our algorithms for fault tolerance with minor modifications. For LP, we modify the *Normalize*() (Algorithm 2, line 8) function. For example, assume that there are two possible candidate paths for flow 0, whose weights are $w[0][0]$ = 10, and $w[0][1]$ = 2.5, respectively, and other candidate paths are with weight $w[0][i]$ = 0, for $i \geq 2$. In LP-FT, we make the *impossible* paths join the candidate path set by modifying their weights, (in this example, let $w[0][2]$ = 2). This leads to the probabilities of choosing path 0, 1, and 2 are 69.0%, 17.2% and 13.8%, respectively. This method activates more paths, and powers on more switches for fault tolerance. For SA-FT, we let additional new core switches join the *best* mapping computed by SA, and change a randomly-chosen host to map to the new core switch.

Our framework combines GBA with LP and SA, respectively. The LP based method and its variants run fast, while the SA method and its variants achieve more energy savings and maintain better FCTs, as demonstrated in the next section.

IV. Simulation Evaluations

This section describes evaluations of our system framework. The goal of these tests is to determine the energy-efficiency and flow performance under various traffic patterns and various network utilizations. The efficiencies and fault-tolerance of algorithms are also evaluated.

A. Simulation Setups

We simulated our system framework on a laptop with an Intel Core i5-2410M 2.30Ghz CPU and 4GB RAM. All of the algorithms are implemented in Java.

Our simulator captures the flow-level events like flow arrivals, departures and transmission rate calculations. Existing packet-level simulators such as *ns*-2 become extremely slow or even impossible when the number of network nodes becomes huge. The traffic is generated for a range of communication patterns at the granularity of network flow which follows Pareto distribution with mean size 50KB. In our simulations, time is split into slices. At each time slice, it updates flow rates and generates new flows if needed. Periodically it calls the scheduler to reassign flows to new paths and changes switch states. For comparison, we also implement the *Greedy Bin Packing* (GBP) method [13], which evaluates possible paths and chooses the leftmost one with sufficient capacity for each flow. When calling the scheduler algorithm (LP, SA or GBP), the simulator also calls GBA for bandwidth allocation. When updating flow rates, we also implement the TCP with slow start and AIMD, and D3 [22] for flow performance comparisons. Our simulator has similar implementation as that in [3], which matches testbed performance very well.

We use Fat-tree topology with 320 switches and 1024 servers. Similar to previous works [2], [3], our synthetic traffic patterns include:

- *Random*: Each server sends to a random destination. Multiple servers can send to the same receiver.
- *Random bijection*: Each server sends to a random destination. Each host receives data from only one sender.
- *Random nonpod*: Each server sends to a random destination not in the same pod as itself. Multiple servers can send to the same receiver.
- *Stride*-512: We number the servers in our Fat-tree topology from left to right, as the leftmost is 0 and the rightmost is 1023. By Stride-512, we mean the server i sends to server (i+512) $\bmod$ 1024.

For a single switch power function (Equation (5)), we set P_u^{cons} = 200 watts, and for each link power function σ = 0, $\mu = 2 \times 10^{-6}$ watts/(Mbps)^2 and α = 2, which are adopted from [19]. Consequently, the maximum power consumption of each switch is 232 watts.

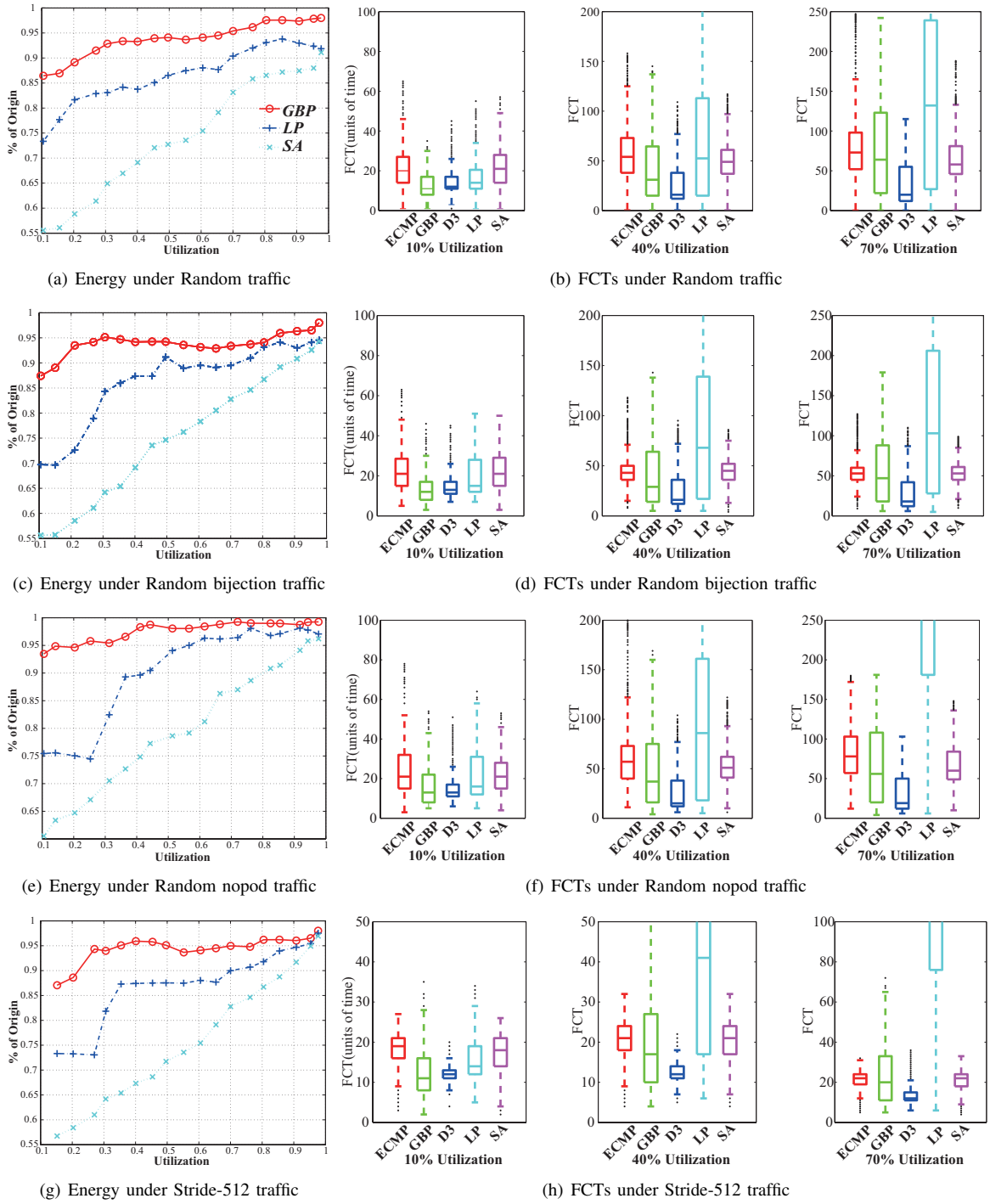
B. Simulation Results

We now explore the efficiency of our framework. The primary metrics include (i) FCTs, and (ii) the ratio of energy consumption with our algorithms over the energy consumption with the always-on baseline.

1) Synthetic demands, varying loads: Energy savings and flow performance heavily depend on the traffic patterns and network utilizations.

Energy saving evaluations: Figs. 4(a), (c), (e), (g) show the energy savings in the Random, Random bijection, Random nonpod, Stride-512 traffic patterns, respectively. In each traffic pattern, we vary the network utilization from 10% to nearly 100% by adding more flows. And for each utilization and each traffic pattern, we run the simulation for 60 seconds, and measure the average energy consumptions during the middle 40 seconds. In all four traffic patterns, SA saves more energy than GBP and LP. In detail, 25% and 8% more energy (at least) can be saved by SA compared with GBP and LP model at low utilization ($10\% - 30\%$), respectively. While the gaps in energy saving between different algorithms decrease as the utilizations grows, SA could save 18% and 7% (on average) more energy than GBP and LP model at medium utilization ($30\% - 60\%$), respectively. When the utilization is close to 100%, all the switches must remain active, and thus all algorithms have similar performance. From another point of view, traffic patterns also affect energy savings. At 20% utilization, SA achieves 42% and 35% energy savings in Random and Random nonpod traffic patterns, respectively, implying that at the same network utilization, the less flows through core switches, the more energy saving.

Flow performance evaluations: In this section, we evaluate the flow performance of different algorithms under synthetic



(a) Energy under Random traffic

(b) FCTs under Random traffic

(c) Energy under Random bijection traffic

(d) FCTs under Random bijection traffic

(e) Energy under Random nopod traffic

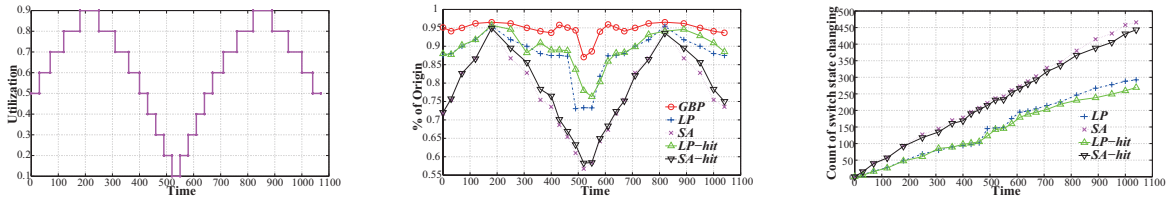
(f) FCTs under Random nopod traffic

(g) Energy under Stride-512 traffic

(h) FCTs under Stride-512 traffic

Fig. 4. Energy consumptions and FCTs under synthetic traffic patterns

traffic patterns. Here, we choose three utilizations of 10%, 40% and 70% to represent low, medium and high utilizations, respectively, and focus on the FCTs. The ECMP and D3 schemes run within the always-on network configuration, while SA, LP and GBP run within subsets of the network. Figs. 4(b), (d), (f) and (h) plot the percentiles FCTs (1^{st}-25^{th}-50^{th}-75^{th}-99^{th}) at different traffic patterns and network utilizations. At low utilization, the median FCTs for all approaches are comparable, while for the 99^{th} percentile FCT, SA and GBP are comparable, but LP performs worse than the two algorithms. As network utilization increases, LP model becomes worse than SA and GBP, because its probabilistic path selection leads to aggressive flow congestions. At medium load, the median FCT of SA is, on average, 11% higher than that of GBP, while the 99^{th} percentile of SA is at least 33% lower than that of GBP. The gaps between SA and GBP



(a) Utilization (b) Energy consumptions (c) Count of switch state changing

Fig. 5. Energy consumptions with sine-wave utilization

TABLE I
THE RATIOS OF ENERGY CONSUMPTION WITH SA BY VARYING ITERATIONS

SA Iteration	Random			Random bijection			Random nopod			Stride-512		
	10%	40%	70%	10%	40%	70%	10%	40%	70%	10%	40%	70%
50	0.594	0.726	0.860	0.581	0.720	0.828	0.618	0.781	0.913	0.557	0.681	0.856
100	0.570	0.703	0.840	0.562	0.703	**0.819**	0.609	0.766	0.887	0.557	0.673	0.832
500	0.573	0.695	0.831	0.560	0.708	0.820	0.605	0.770	0.878	**0.550**	0.690	0.827
1000	**0.563**	0.702	0.838	0.554	0.696	0.822	0.610	0.765	0.891	0.564	**0.668**	0.831
5000	0.567	**0.691**	0.834	0.562	0.695	0.821	**0.604**	0.772	0.875	0.551	0.678	0.823
10000	0.565	0.695	**0.830**	**0.551**	**0.691**	0.825	0.605	**0.761**	**0.871**	0.560	0.670	**0.818**

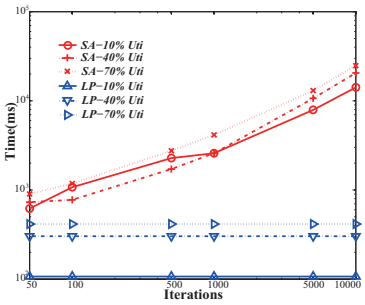
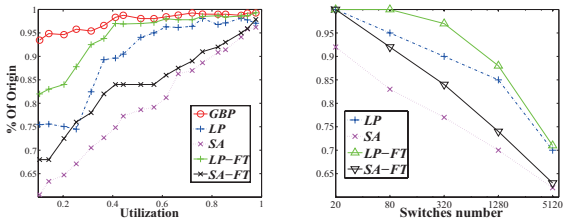


Fig. 6. Time consumptions by varying iterations



(a) Varying utilizations (b) Varying scales

Fig. 7. Energy consumptions with fault tolerance

become larger in terms of the 99^{th} percentile, but become closer in terms of the median at high utilization.

2) Diurnal variation demand: As the network utilization varies with time in product data centers, we capture this demand by simulating a sine-wave utilization which is inspired by [13]. As Fig. 5(a) shows, the highest utilization is 90%, while the lowest is 10%. The simulation runs for 40 seconds and 80 seconds at low and high utilization, respectively, to imitate the diurnal utilization pattern. We compare energy-efficiency of LP and SA, together with LP-hit and SA-hit. From measurements of the half of the wave length, we record and calculate the energy consumptions from the 200^{th} time tick to the 550^{th} time tick, which demonstrate that SA and LP save 18.4% and 5.6% more energy than GBP, respectively. While SA-hit consumes more energy than SA, which is about 3% in this load decreasing phase, it consumes almost equal energy as SA in the load increasing phase. This phenomenon also appears in LP-hit, as illustrated in Fig. 5(b). Since *Hit mechanism* indeed incurs more energy, it could reduce the number of switch state changing (Fig. 5(c)). We also note that SA leads to more state changing times than LP, which we believe, is the expense for higher energy savings.

3) Iterations and running times: To explore the energy savings of SA under different iterations, we conduct the experiments using synthetic traffic patterns and various utilizations as before. These results (Table I) conform with our expectation that more iterations create *better* state that can be used to save more energy. Fortunately, most of the performance improvements appear in the first few iterations. Next, Fig. 6 shows the running times of SA for different iterations, compared with LP under various utilizations. These results report that the running times of SA with few hundreds of iterations are not much longer than LP.

4) Fault tolerance: We use the Fat-tree topology as before, and the Random traffic pattern as the underlying traffic. For LP-FT, we add two more possible paths into the candidate path set for each flow, while adding one more core switch (if exists) for each pod to the best state for SA-FT. Fig. 7(a) shows that additional energy required by fault tolerance decreases as the utilization grows. Furthermore, as the network scale increases, the cost of fault tolerance decreases, which is confirmed by the results of Fig. 7(b) where the number of switches changes from 20 to 5120.

In summary, our SA and LP methods together with GBA show better energy-efficiency than GBP algorithm under the synthetic traffic patterns. In terms of flow performance, LP

impacts the FCT aggressively. In contrast, SA shows comparable median FCTs but lower 99^{th} percentiles compared with GBP. Besides, the running time of SA is not much longer than LP since its improvements almost always appear in the first few iterations. We also demonstrate that our methods can be easily extended for fault tolerance and the incurred energy consumptions are controllable.

V. Related Works

Energy-efficient DCNs: Many approaches have been proposed on improving the energy-efficiency of DCNs. The first type of these works is to design novel architecture. [15] proposed a server-centeric data center structure that conserves energy by varying bandwidth availability based on traffic demand. The second type is to optimize energy-efficiency of DCNs by traffic engineering methods. Heller *et al* [13] presented *ElasticTree*, a network-wide power manager, which dynamically adjusted the set of active network elements. But *ElasticTree* assumes that a complete prior knowledge of incoming traffic is known. Wang *et al* [21] proposed CARPO, a correlation-aware power optimization algorithm that dynamically consolidates traffic flows onto a set of switches and shuts down unused network devices. Andrews *et al* exploited speed scaling [4] and power-down [5] techniques to route and schedule continuous flows, but the transmission speed for each flow was given as a constant.

Flow-level optimization: The performance of OLDI applications heavily depends on FCTs. There is abundant work that deals with the subject of data center transport designs. D3 [22] first introduced deadline information combined with explicit rate control. PDQ [14] showed that minimizing FCTs requires preemptive flow scheduling.

There are few works that improve DCN energy-efficiency and performance simultaneously. Wang *et al* [20] proposed a novel energy-saving model for data center networks by scheduling and routing deadline-constrained flows, based on speed scaling and power-down strategies. But the power function of switches they used was only based on links (or ports). We use the general switch power function and propose approaches that could make energy consumptions of DCNs approximately in proportion to their traffic loads, while the flow-level performance degradation is guaranteed.

VI. Conclusion

In this paper, we studied the energy-efficiency of DCNs, where the flow-level performance was guaranteed. The key idea of our framework was that we decoupled the path selection and rate allocation for flows. In path selection, we assigned flows to paths with fewer switches to minimize network energy consumptions. In rate allocation, we efficiently utilized link bandwidth to satisfy flow demands which were estimated with implicit deadlines. With these approaches, we were able to achieve both energy efficiency and better flow performance compared with existing works.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61472181, 61321491, 61502224, 91218302; China Postdoc Science Fund under Grant No. 2015M570434; Jiangsu Natural Science Foundation under Grant No. BK20151392; Jiangsu Key Technique Project (industry) under Grant No. BE2013116; EU FP7 IRSES MobileCloud Project under Grant No. 612212. And this work is also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] America's data centers consuming and wasting growing amounts of energy. http://www.nrdc.org/energy/data-center-efficiency-assessment.asp.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM 2008*, pages 63–74.

[3] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI 2010*, pages 19–19.

[4] M. Andrews, A. F. Anta, and L. Zhang. Routing for power minimization in the speed scaling model. *IEEE/ACM TON*, 20(1):285–294, 2012.

[5] M. Andrews, A. Fernández Anta, L. Zhang, and W. Zhao. Routing and scheduling for energy and delay minimization in the powerdown model. *Networks*, 61(3):226–237, 2013.

[6] L. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *Micro, IEEE*, 23(2):22–28, March 2003.

[7] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM 2010*, pages 92–99.

[8] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *FOCS 1975*, pages 184–193.

[9] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM 2008*, pages 68–73.

[10] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. In *ACM SIGCOMM 2009*, pages 51–62.

[11] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM 2009*, pages 63–74.

[12] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM 2008*, pages 75–86.

[13] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *NSDI 2010*, pages 249–264.

[14] C.-Y. Hong, M. Caesar, and P. Godfrey. Finishing flows quickly with preemptive scheduling. *ACM SIGCOMM 2012*, pages 127–138.

[15] L. Huang, Q. Jia, X. Wang, S. Yang, and B. Li. Pcube: Improving power efficiency in data center networks. In *CLOUD 2011*, pages 65–72.

[16] P. Mahadevan and P. Sharma. A power benchmarking framework for network devices. In *NETWORKING 2009*, pages 795–808.

[17] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. Minimizing flow completion times in data centers. In *INFOCOM 2013*, pages 2157–2165.

[18] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). *ACM SIGCOMM 2012*, pages 115–126.

[19] L. Wang, F. Zhang, J. Arjona Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu. Greendcn: a general framework for achieving energy efficiency in data center networks. *IEEE JASC*, 32(1):4–15, 2014.

[20] L. Wang, F. Zhang, K. Zheng, A. V. Vasilakos, S. Ren, and Z. Liu. Energy-efficient flow scheduling and routing with hard deadlines in data center networks. In *ICDCS 2014*, pages 248–257.

[21] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao. Carpo: Correlation-aware power optimization in data center networks. In *INFOCOM 2012*, pages 1125–1133.

[22] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: Meeting deadlines in datacenter networks. In *ACM SIGCOMM 2011*, pages 50–61.

Blending photons with electrons to reduce the energy footprint of IPTV networks

Fernando M. V. Ramos*, Jon Crowcroft†, Ian H. White†
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal* University of Cambridge, UK†

***Abstract*—The rapid growth of IPTV services and the resulting increase in traffic volumes is raising concerns over energy consumption. In this paper we propose to save energy by shifting particular IPTV traffic from power-hungry electronic routing to greener optical switching. The traffic profile of IPTV results in such a hybrid switching approach to allow *both* energy and bandwidth efficiencies. To achieve this goal we designed a novel protocol that allows the use of optical bypass in IPTV networks. By means of a trace-driven analysis of a large dataset we demonstrate the energy efficiencies obtained to be substantial, reaching power savings of over 40% under normal load conditions. This result represents a four-fold increase in energy efficiency when compared with recent proposals.**

I. Introduction

The past few years have witnessed the rapid roll-out of IPTV services. IPTV has been launched by major service providers worldwide [1] and its popularity is on the rise. According to the most recent report from Digital TV Research [2], in the period from 2011 to 2013 the number of IPTV customers has increased from 36 to over 90 million, and is expected to reach 191 million by the end of 2020.

IPTV is a resource intensive service with stringent quality of service requirements. Each video stream is encoded at a bit rate that can vary from around 4Mbps (SDTV) to 20 Mbps (HDTV). In the future this figure may increase by one or two orders of magnitude, with the advent of ultra high definition video standards (4K, UHDTV). Besides the resulting increase in bandwidth requirements, the number of TV channels offered is also expected to grow. Current IPTV operators already offer near one-thousand TV channels [3] to its customers. But recent trends anticipate the likely growth of the number of TV channels in the near future. Narrowcasting services – broadcasting to a very small audience [4] –, for example, are growing in importance.

The increase in bandwidth required to support an increasing number of users and of TV content has led to concerns about the energy consumed by the infrastructure. Various studies have highlighted the effects of GreenHouse Gases (GHG) emissions and their consequences on climate change [5] and on the economy. ICT represents an important source of energy consumption and of GHG emissions, with 37% of the total ICT emissions due to the network infrastructure [6].

In this paper we focus our attention in the energy consumption of traditional (push-based) linear TV over IP networks. In this type of system TV programs are broadcast on the different channels according to a known schedule (content is therefore *pushed* to users). This type of service is fundamentally different from pull-based approaches such as time-shifted TV[1] or VoD. Linear TV broadcasts do not have the natural scaling properties of time-shifted TV [7], increasing the challenge of improving the efficiency of these networks. For instance, the intrinsic nature of linear TV services precludes the use of energy optimisation approaches based on caching techniques, as recently-proposed for time-shifted TV [7], [8]. Alternative techniques are therefore needed.

Today, IPTV systems[2] are inefficient. In current deployments *all* TV channels are distributed to *all* local routers, despite particular channels having no viewers at particular time periods. Considering this inefficiency problem, we have proposed a scheme [9] – *selective pre-joining* – where only a selection of TV channels is distributed in the network, instead of all. We have shown, based on real data, that by using this scheme it is possible to save bandwidth and energy while affecting a very small number of channel switching requests. For instance, we have demonstrated that if core routers pre-joined only the channels that have viewers the effect in the quality of service would be residual (less than 0.1% channel requests would be affected) while achieving important efficiency gains.

In this paper we propose to go further in terms of energy-efficiency by following a different approach. The technique we propose is based on the introduction of optical switching in the network. The rationale is the fact that optical switching techniques are more energy-efficient than their electronic counterpart [10]. In particular, we assess the opportunities for performing *optical bypass* in IPTV networks, and propose a novel protocol for this purpose. With optical bypass, traffic not destined for a given network node is not processed electronically by that node. This traffic is all-optically switched.

By avoiding electronic processing and performing optical switching instead, energy savings are to be expected. We demonstrate in this paper this hypothesis to be true. For this purpose, we evaluated our protocol by means of trace-driven analyses using a dataset from an operational IPTV service provider. The dataset scales up to 150 TV channels, six months, and 255 thousand users. We demonstrate that by using the proposed scheme IPTV service providers can significantly reduce the energy consumption of their networks. For instance, for normal traffic load conditions our proposal presents a four-fold increase in energy-efficiency when compared with *selective pre-joining* [9].

[1] On-demand access to previously broadcast TV content, also known as catch-up TV.

[2] We will henceforth use IPTV to refer to linear TV services distributed over an IP network.

ISBN 978-3-901882-83-8

II. Background

A. IPTV networks

A traditional "walled garden" IPTV network can be split logically into three main domains – the access network, the metropolitan network, and the IP network. The IP network usually has a two-level, hierarchical structure [11]: the regional network (sometimes called edge) and the core (Figure 1).

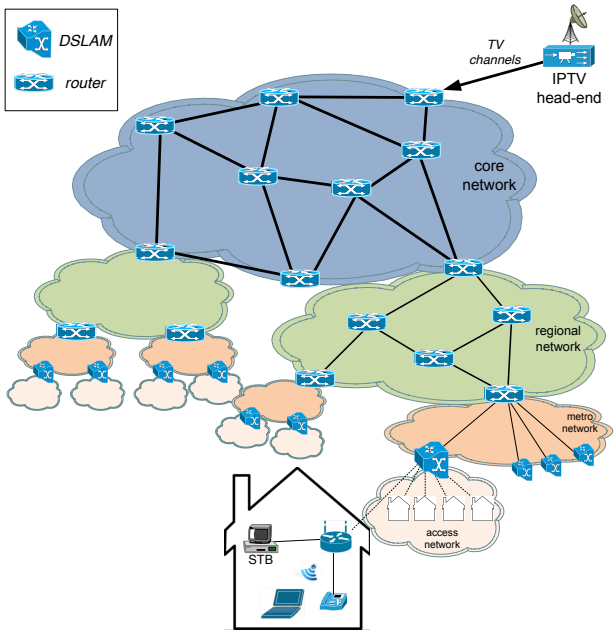


Fig. 1. IPTV network

In an IPTV system, live TV streams are encoded in a series of IP packets and delivered through the network to the residential broadband access network. The core network comprises a small number of large routers in major population centres. The core routers are often highly meshed, with high-capacity WDM fibre links interconnecting them. In the regional network, routers are normally lower-end routers with high port density, where IP customers get attached to the network. These routers aggregate the customer traffic and forward it toward the core routers. The metro network serves as the interface between the regional network and the access network. The access network connects each home to one of the edge switches in the provider's network. There is a wide variety of access technologies: from ADSL (Asymmetric Digital Subscriber Line) to fibre-based solutions (FTTx). Inside the household, a residential gateway connects to a modem and one or more Set Top Boxes (STBs). Finally, each STB connects to a TV.

The TV channels are distributed from the TV head-end to edge nodes (DSLAMs in Figure 1) through bandwidth-provisioned multicast trees. Current networks use *static* IP multicast within a single network domain. By static multicast we mean all receivers are known beforehand, and no new group members are allowed to join – we have a *static set of receivers* for all TV content. Again referring to Figure 1, this means *all* DSLAMs join *all* multicast groups (thus receive content from *all* TV channels). This occurs despite the fact that particular channels have no viewers at particular time periods [9].

B. Core optical IP networks

An optical IP network can be seen as being made up of two layers, the IP layer and the optical layer. This is shown in Figure 2. In the first generation of optical networks, all the lightpaths incident to a node had to be terminated, i.e., all the data carried by the lightpaths would be processed and forwarded by IP routers. This is represented in the figure by lightpath 1. This wavelength is OEO (Optical-Electrical-Optical) converted at each node. In contrast, the new generation of optical networks includes elements such as the Optical Cross Connect (OXC) which allow some lightpaths to bypass the node. This approach allows IP traffic whose destination is not the intermediate node to directly bypass the intermediate router via a cut-through lightpath. This is represented by lightpath 2. This wavelength bypasses all nodes.

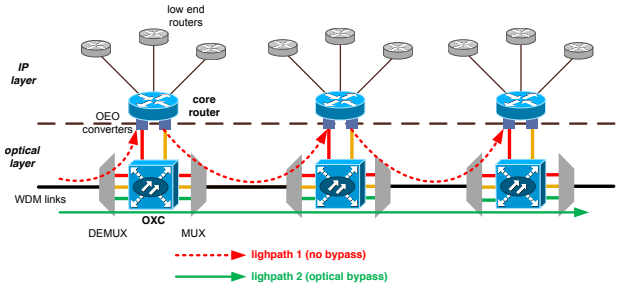


Fig. 2. Optical network employing optical bypass techniques

Several researchers have pointed out that optical bypass technology is one important method to reduce the power consumption of IP networks [10], [12], [13]. This technique can save energy because it can reduce the total number of active IP router ports, and these play a major role in the total energy consumption of an optical IP network [13]. Shifting traffic from power-hungry routers to low-power optical switches by means of optical bypass is therefore an effective technique to save energy in optical networks.

III. Optical bypass of popular TV channels

With the goal of reducing energy consumption, in this paper we consider the introduction of energy-friendly optical switching techniques in the core of optical IPTV networks. In particular, we propose the introduction of optical bypass. With optical bypass traffic not destined for a given IP router is placed onto a wavelength that is not processed by that router. Instead, this traffic is all-optically switched. Due to the circuit-switching nature of optical networks, however, only *long-lived flows* can be considered realistic targets for optical bypass.

Conveniently, some IPTV traffic is in this category. Some TV channels are very popular, having viewers everywhere in the network, at any particular time. Optically switching such long-lived flows can therefore be advantageous energy-wise. Other less popular and niche channels have periods without any viewers in particular locations, so it is wasteful to distribute them continuously everywhere. The dynamic nature of electronic packet-switching nodes is therefore ideal to switch this type of traffic. This guarantees the network is bandwidth efficient, by allowing these TV channels to be quickly removed from or added to the network as needed.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

A. Protocol for optical bypass in IPTV

Considering the above, we propose a protocol to be used in the core of IPTV distribution networks, blending electronic routing with all-optical switching. We assume the network core to be composed of hybrid nodes, each including a multicast-capable WDM optical cross connect (OXC) and a multicast-enabled IP router, as illustrated in Figure 2. The inclusion of the OXC between the input ports and the router allows optical bypass to be performed. We further assume these nodes to be GMPLS-capable. A unified control plane such as GMPLS allows the integration of optical circuit-switching techniques with electronic packet-switching. The main idea of the scheme is for popular TV channels to be all-optically switched (switched at the optical layer), while the rest are electronically routed (switched at the IP layer). The network distributes the two different groups of channels in two (disjoint) sets of wavelengths. The wavelengths from one set optically bypass the nodes, whereas the other wavelengths are sent to the routers for processing. We restrict the use of the proposed scheme to the optical network core, as this is the only location where it is realistic to assume the presence of OXC equipment in the medium-term.

Algorithm 1 Processing at the IPTV source

```
1:  while true do
2:      sleep(Δτ)
3:      send_to_core-reg_nodes(ACTIVE_CHANNELS_REQ)
        {Wait until all requests are received...}
4:      CPop ← ALL_TV_CHANNELS
5:      CNonPop ← ∅
6:      for i = 1 to NUMBER_OF_NODES do
7:          CPop ← CPop ∩ ActiveCh[i]
8:      end for
        {CPop now includes all popular TV channels}
9:      for i = 1 to NUMBER_OF_NODES do
10:         CNonPop ← CNonPop ∪ (ActiveCh[i] ∉ CPop)
11:     end for
        {CNonPop now includes the other TV channels with
        viewers}
12:     λ_o ← [Wavelengths filled with CPop channels]
13:     λ_e ← [Wavelengths filled with CNonPop channels]
14:     send_to_all_nodes(SWITCH_CHANGE_REQ, λ_o, λ_e)
15: end while
```

Algorithm 2 Processing at each core-regional node

```
1: while true do
2:     MESSAGE = msg_rcv_from_source()
3:     if MESSAGE == ACTIVE_CHANNELS_REQ then
4:         ActiveCh ← get(McastFwdTable)
5:         send_to_source(ActiveCh)
6:     end if
7: end while
```

The protocol for optical bypass in IPTV networks consists of three algorithms. Algorithm 1 runs at the IPTV source, Algorithm 2 runs at core-regional nodes (Figure 1), and Algorithm 3 runs at the core nodes (including core-regional ones). The details of the proposed protocol follows.

Step 1. After a specified time interval, $\Delta\tau$, the source transmits a message requesting all hybrid core-regional nodes to submit their active channels (algorithm 1, lines 2-3). An *active channel* is a channel for which there is at least one viewer. This message sent by the source serves as a trigger for all core-regional routers to send this information back to the source as soon as possible. Considering that all nodes are GMPLS-capable, this information can be sent as a TE Notify message. RSVP-TE Notify messages were added to RSVP-TE[3] to provide general event notification to nonadjacent nodes.

Step 2. Each regional-core node then sends information on its *active channels* to the IPTV source. As the active channels are those being distributed by the regional-core router to its region, the multicast forwarding table of this router contains a line with their multicast group addresses and the interfaces used to forward packets to. The information requested can thus be easily retrieved and sent back to the source (algorithm 2, lines 3-6). Again, an RSVP-TE Notify message can be used.

Step 3. Once the source receives these sets from all routers, it checks which TV channels should be optically switched (the popular ones), and which should be electronically routed (the remainder channels with viewers). The popular channels are those which have viewers everywhere. Their multicast group addresses are present in the multicast forwarding tables of every core-regional router. The intersection of all sets received by the source thus results in a new set with the list of popular channels[4] (algorithm 1, lines 6-8). The union of the active channels of each set which are not popular results in a set with the non-popular TV channels (algorithm 1, lines 9-11).

Step 4. The TV channels are distributed, from the source, in two distinct sets of wavelengths: λ_o and λ_e. The popular channels are distributed using N different wavelengths: $\lambda_o = N \times \lambda$. The others are sent in a disjoint set of M different wavelengths: $\lambda_e = M \times \lambda$. The number of wavelengths in each set depends on the number of TV channels and its bit rate, and on the capacity of each wavelength. The IPTV source decides the composition of each set of wavelengths and informs all core nodes of its decision (algorithm 1, lines 12-14). This information can be sent in the form of an RSVP-TE PATH message. This is one of the messages used to allocate resources in the network. In multicast scenarios, only one PATH message

Algorithm 3 Processing at each core node

```
1:  while true do
2:      MESSAGE = msg_rcv_from_source()
3:      if MESSAGE == SWITCH_CHANGE_REQ then
4:          for all λ ∈ λ_o do
5:              switch_optically(λ)
6:          end for
            {Wavelengths in the set λ_o are optically bypassed}
7:          for all λ ∈ λ_e do
8:              route_electronically(λ)
9:          end for
            {Wavelengths in the set λ_e are sent to the router}
10:     end if
11: end while
```

[3] As its name implies, the Resource Reservation Protocol - Traffic Engineering (RSVP-TE) is an extension of the resource reservation protocol (RSVP) for traffic engineering, and is used as part of the GMPLS control plane for this purpose.

[4] We are abusing the term "popular" in this paper. If one TV channel has a single viewer in each region then it is included in the popular set. We use this term to ease the understanding of the scheme.

needs to be sent to multiple receivers, thus conserving network bandwidth.

Step 5. Each core node then sets up its switching state to optically switch the λ_o group (these wavelengths will therefore optically bypass the routers), and electronically route the λ_e group (algorithm 3, lines 3-10).

IV. Methodology and Dataset

The research community working on IPTV systems has relied upon hypothetical user models which are sometimes different from reality and can lead to incorrect estimation of system performance. Constant-rate Poisson models are generally used as workload model for these systems. Examples include [14], [15], among several others. Unfortunately, these models does not capture IPTV user behaviour well. For instance, users switch channels more frequently than these simple models predict. This fact was proved by Qiu *et al.* [3]. The authors made a comprehensive analysis of real data from an operational nationwide IPTV system (AT&T) where they show that the simple mathematical models generally used are not good. Faced with this concern, the solution we propose in this paper is evaluated by means of trace-driven analysis of a large dataset from an IPTV provider.

The analysis of our dataset led us to the same conclusions as in [3]. In Figure 3 we exemplify one of the problems of using a simple Poisson distribution as a mathematical model to represent the behaviour of IPTV users. The figure presents the Cumulative Distribution Function of the number of channel switches during one-minute periods (a *zapping period*, according to [1]). The analysis was done on the entire dataset we describe in Section IV-A. In the figure we compare the empirical data with a Poisson distribution with parameter λ equal to 1.948 (the one that fits better the empirical data).

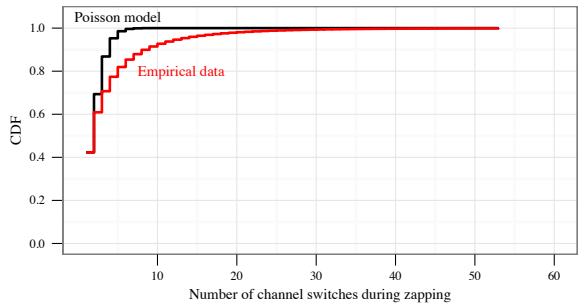


Fig. 3. Number of channel switches in zapping mode

As can be seen, the Poisson model is conservative in terms of the number of channel switches a user performs during zapping periods. For example, the probability of a user making five channel switches or more in a one-minute period is negligible when using the Poisson distribution. But in fact by observing the empirical data one can conclude that there is a 20% probability of a user switching channels five times or more during a zapping period.

A. Dataset

We obtained a collection of IPTV channel switching logs from an IPTV service offered by an operational backbone provider. This is a commercial, nationwide service, offering 150 TV channels over an IP network. The access links use ADSL technology and the network is composed of 680 DSLAMs distributed along 11 regions. To give an idea of the scale of the dataset, the 700GB trace spans six months and records the IGMP messages on the channel changes of around 255 thousand users. The number of daily channel switchings clocks 13 million on average. Table I summarises these statistics.

TABLE I. Dataset statistics

Trace duration	6 months
Number of users	255 thousand
Number of DSLAMs	680
Average number of daily channel switching events	13 million
Size of the dataset	700 GB

B. Validation of the dataset

To assure the representativeness of our dataset and the evaluation that ensues, we compared the results from Qiu et al [3] with the results from the analysis of our dataset (the two datasets are from different IPTV services offered in different countries). For validation we analysed the number of online users during the course of representative weeks, and found the same very strong diurnal patterns as Qiu *et al.* [3]. We also examined the long term distribution of channel popularity and found the same high skewness of popularity, which can be modelled using Zipf-like distributions. Finally, we also conclude that IPTV users switch channels very similarly in both studies. By analysing the entire dataset we observed that 55% of all channel switching was linear, up or down to the next or previous TV channel (in constrast to more targetted switching). Qiu *et al.* [3] reported 56% in the AT&T dataset. The full detail of this validation is out of scope of this paper. We leave an in-depth analysis as future work.

V. Evaluation

For the reasons explained before, the scheme we propose is evaluated by means of a trace-driven analysis. The IPTV trace detailed in Section IV-A is used as input to the analysis performed. All results we present next arise from the analysis of the entire data set (6 months, 255 thousand users). The evaluation is threefold. First, we investigate the scalability of the protocol. Second, we analyse the opportunities for optical bypass when running the proposed protocol in the network under study. Finally, we analyse the impact the use of this protocol has in power consumption of the IPTV network.

A. Scalability

For a network protocol to be scalable it is important that it does not impose a significant processing overhead to the network nodes and that it does not add a great amount of signalling traffic to the network. By guaranteeing a relatively long update interval for the control information (the $\Delta\tau$ variable in the proposed protocol) it is possible to guarantee a low overhead to the nodes and to the network as a whole. On the other hand, to assure the best performance it is important that the network state[5] is consistent with network usage (in this

[5]In this context, the network state consists of the wavelength switching configuration at each node, and the set of TV channels transported in each wavelength group, λ_o and λ_e.

particular case, it should reflect channel popularity). Having a short update interval marries with this objective.

It is known that channel popularity is relatively stable over short time frames, and that it becomes more dynamic when longer time frames are considered [16]. Regular updates may therefore not be needed. We analyse the henceforth called *TV channel churn rate* in the 11 core-regional nodes of this network to attest this. We compare the active TV channels at time τ with the active channels at time $\tau + \Delta\tau$, for different values of $\Delta\tau$. The number of channels that are different between the two sets in two consecutive periods is the TV channel churn rate. The results are shown in Figure 4, for each region, and for five values of $\Delta\tau$. The median of the channel churn rate over the whole period of the trace (6 months) is presented, with the lower and upper error bars representing the 5th- and 95th-percentile, respectively.

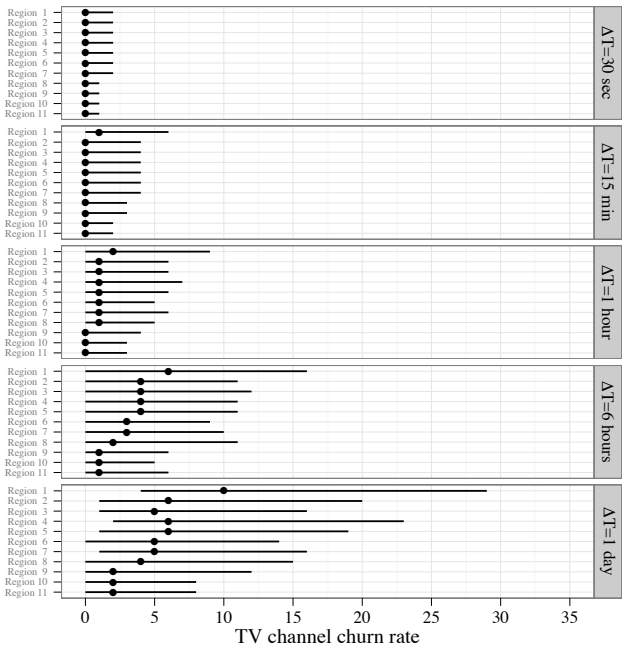


Fig. 4. TV channel churn rate for all eleven regions, for five values of the update interval

By analysing the results in Figure 4, we conclude that the churn rate is usually quite low, particularly for values of $\Delta\tau$ below 1 hour. A long update interval of 15 minutes, for instance, seems a good compromise. It does not represent a significant overhead to the network, while at the same time guarantees that the network state changes with channel popularity dynamics.

B. Opportunities for optical bypass

The protocol proposed divides the TV channels into three groups: the popular channels, the unpopular channels, and the channels without viewers. The channels from the former group optically bypass the routers. Those from the second group are sent for the router for electronic processing. Finally, those from the latter group are not distributed by the IPTV source (when a user switches to a channel without viewers a request is sent to the source for its quick distribution and the channel becomes part of the group of unpopular channels). To understand the opportunities for optical bypass in the core of the IPTV network, we need to quantify how many channels would be included in each group, at regular intervals. For this purpose, we retrieve the number of channels in each set (popular, unpopular, and channels with no viewers), for the entire trace. We consider for the analysis an update interval equal to 15 minutes, for the reasons explained above. This is the periodicity with which we retrieve the number of channels in each set. In Figure 5 we present the results obtained (median, 5th-, and 95th-percentile) from the analysis of the entire dataset.

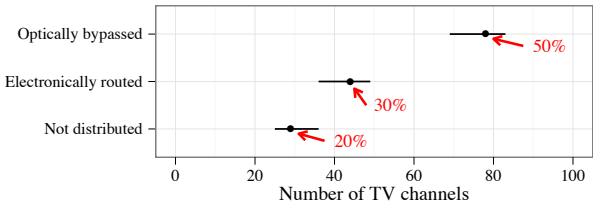


Fig. 5. Average number of TV channels that are optically bypassed, electronically routed and not distributed, respectively

We start the analysis from the bottom of the figure. On average, one fifth of the TV channels do not need to be distributed by the IPTV source. Recall that not distributing this traffic has a negligible impact on the service [9]. The remaining 80% TV channels are distributed to the network core. Around 50% of the TV channels can be optically bypassed. This means that, on average, at any one time, half of the channels have at least one viewer in each region. The number of channels requiring electronic processing can thus be reduced to around 30%.

C. Impact on energy consumption

After understanding that by using the proposed protocol there are clear opportunities to introduce optical bypass in IPTV networks we now analyse the impact this has on energy consumption. By employing this technique energy savings are expected for two reasons. First, some traffic flows (the popular TV channels) bypass some routers. This reduces the number of bits requiring electronic processing, thus avoiding energy-expensive OEO conversions, buffering, and forwarding table lookups. The work is shifted to optical switches, which are at least two orders of magnitude more energy efficient when compared to its electronic equivalent [10]. Second, as TV channels without viewers are not distributed, network load is reduced and even less bits require electronic processing in the routers.

1) Selective pre-joining in core optical networks: Before presenting results from our proposal, we return to the scheme proposed in [9], *selective pre-joining*, and use it as our baseline. However, we consider an optical network scenario, which was not considered by the original study. As in this work we consider a core optical network, we need to integrate the power consumption of the optical layer components into the model. This will allow a fair comparison with the proposal made by Ramos *et al.* [9]. In fact, this refined model will reinforce the effectiveness of *selective pre-joining*.

Several factors affect the power consumption of a core network node [17]. First, the base chassis power. This is the

TABLE II. LINECARD POWER PROFILE

Energy component and description	Estimate from [18].
Power consumed by unconnected linecard card (P_c)	6.936 W
Power consumed per connected Ethernet port (P_E)	1.102 W
Per-packet processing energy (E_p)	197.2 nJ
Per-byte energy (E_b)	3.4nJ

power to maintain the chassis on. It is a fix amount independent of load, including the power consumed by components such as fans, memory, etc. Second, the number of active linecards. A linecard is the electronic circuit that interfaces with the network. Third, the number of active ports in each linecard. Fourth, port capacity. This is the line rate forwarding capacity of individual ports. Fifth, port utilisation. This is the actual throughput flowing through a port, relative to its capacity. Sixth, power consumption of the transponders. In optical networks, associated with each wavelength (port) is a transponder (OEO converter), as was shown in Figure 2. The transponder interfaces the router to a fibre optic cable. Its main function is to perform the required OEO conversions. Considering this, the power consumption model is presented in Equation 1.

$$P = P_{ch} + \sum_{i=0}^{L} P_{l_i} + K_T P_T \tag{1}$$

In this equation P_{ch} refers to the power consumption of the chassis. L is the number of linecards that are active, and P_{l_i} is the power consumption of linecard i. The power consumption of each linecard is calculated based on the model proposed by Sivaram *el al.* [18] for a NetFPGA card, and is presented as Equation 2. By using a high-precision hardware-based traffic generator and analyser, and a high-fidelity digital oscilloscope, the authors devised a series of experiments allowing them to quantify the per-packet processing energy and per-byte energy consumption of such linecard.

$$P_l = P_c + KP_E + N_I E_p + RE_b \tag{2}$$

In this equation P_c is the constant baseline power consumption of the NetFPGA card (without any Ethernet ports connected); K is the number of Ethernet ports connected; P_E is the power consumed by each Ethernet port (without any traffic flowing); N_I is the input rate in packets per second (pps); E_p is the energy required to process each packet; R is the traffic rate in bytes per second (we assume the input rate is equal to the output rate); E_b is the total per-byte energy (this includes the energy required to receive, process and store a byte on the ingress Ethernet interface, and the energy required to store, process and transmit a byte on the egress Ethernet interface). The inputs to this model are presented in Table II, again based on the measurements reported in [18].

Finally, returning to Equation 1, K_T is the number of transponders (one per port) and P_T is the power per transponder. Every time a new port needs to be turned on, a new transponder is also activated. We assume the power consumption for each transponder to be 73 W, based on Alcatel-Lucent WaveStar OLS 1.6T ultra-long-haul systems [19]. This figure has been used in related work [13], [20].

For evaluation we consider the three scenarios presented in Table III: `150SD`, an IPTV service offering of 150 SDTV channels; `700HD`, 700 HDTV channels; and `3kUHD`, 3000 UHDTV channels. The first two scenarios represent current IPTV service offerings, whereas the latter is a futuristic scenario. For the first scenario, we assume a router with four linecards with 4x1Gbps Ethernet ports each. For the second scenario we scale up the node to sixteen linecards of the same type, for it to be able to handle the increased aggregate throughput. The capacity of each node is now assumed to be equal to 64Gbps. The capacity of the nodes of the third scenario has to scale up to the Tbps range. We assume fourteen 4x40Gbps linecards for an aggregate capacity of 2.2Tbps. This is a different type of linecard from the one measured by Sivaram *et al.* [18]. We therefore assume a 4x40Gbps linecard presents the same power profile as fourty 4x1Gbps.

TABLE III. DESCRIPTION OF THE THREE SCENARIOS

Scenario	Media format	Bit rate	TV channels	**Bandwidth savings**
150SD	SDTV	4 Mbps	150	**0.3 Gbps**
700HD	HDTV	20 Mbps	700	**7 Gbps**
3kUHD	4K	200 Mbps	3000	**300 Gbps**

The results we present first illustrate the relative power savings of using the *selective pre-joining* scheme as a factor of the baseline traffic load, according to equation 3. The baseline traffic load is the load of a node that does not use the scheme. This load obviously includes IPTV traffic.

$$\frac{P\,(baseline) - P\,(selective\_joining)}{P\,(baseline)} * 100 \tag{3}$$

In Equation 3, $P\,(baseline)$ is the power consumption at baseline traffic load, whereas $P\,(selective\_joining)$ is the power consumption when using the proposed scheme (a lower value due to the decrease in IPTV traffic). In the figures we present results for baseline load values varying from 25% to 75%.

In accordance to the results presented in the previous section (Figure 5), we assume that only 20% of the channels are not distributed to the core. The results we present in Figure 6 thus correspond to a reduction of IPTV traffic in the network core to 80%. In this figure the lines labeled *(IP only)* are the results without considering optical components in the power consumption model (as per [9]), whereas *(IP + opt)* are the results using our augmented model, considering the optical transponders.

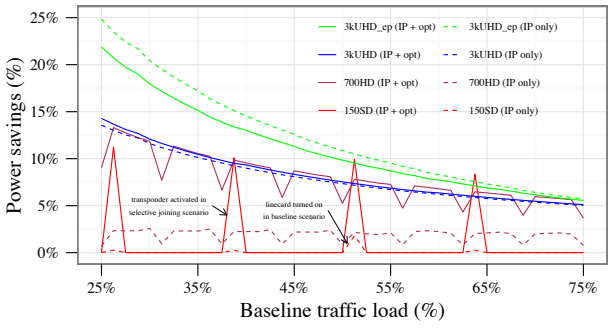


Fig. 6. Power savings of using the selective pre-joining scheme proposed by Ramos et al [9] considering an optical IP network (core network)

As can be seen, by considering the optical ports the results change significantly. The main reason is the fact that the transponders are power-hungry equipment. This results in an increased advantage in using the *selective pre-joining* scheme in some scenarios, as reducing traffic load decreases the number of active transponders. It is particularly relevant to mention scenario `700HD`, which is typical in current networks (this scenario is based on AT&T's IPTV service offering [3]). The use of the scheme proposed in [9] increases the power savings to around 10% under normal traffic loads [11] when we consider optical components.

One aspect that deserves explanation is the lines in the plots not being completely smooth (the little "steps"). This is particularly evident in the first two scenarios, `150SD` and `700HD`. The reason is that the *x*-axis represents the baseline traffic load in the node (without *selective pre-joining*), while the power savings arise from the new traffic load (with *selective pre-joining*) being lower. The power saving peaks that appear in the graph represent transition points, when a particular event that increases significantly the energy consumption occurs: when an additional linecard needs to be turned on or the activation of another transponder (as this component consumes more power than a linecard, the peaks are more pronounced). For instance, in the `150SD` *(IP only)* scenario there is a peak precisely in the middle of the plot. This is because a 50% load in that scenario represents a data rate equal to 4Gbps. At this point, the network node has to turn on a new linecard (recall that we are assuming 4x1Gbps linecards). With *selective pre-joining* the network load would be lower than the baseline traffic load, a bit under 4Gbps. So the linecard does not need to be turned on yet. While the traffic load does not increase over that transition point the proposed scheme therefore presents a higher-than-average power saving advantage. In the `700HD` scenario the same occurs, but more frequently. This is due to the fact that in this scenario the network nodes have eight times more linecards, so the effect occurs eight times more than in the `150SD` case. A similar effect occurs in the futuristic scenario. But, as the baseline power consumption is much higher than in the first two, the bumps are less pronounced, and are hence imperceptible in the figure.

We now turn to the line in the plot we have not mentioned yet: `3kUHD_ep`. As is well known, current network equipment is not energy proportional [21]. The baseline power (from maintaining the chassis powered on) is very high and is, by a large margin, the main component of router power consumption. In the future it is expected network equipment to increasingly present a more energy proportional profile, so in the plots we also include, for the futuristic scenario `3kUHD`, the situation where all routers are energy-proportional (EP) (`3kUHD_ep`). As can be observed, using the *selective pre-joining* scheme leads to a higher relative gain considering that different starting point in the analysis (i.e., the use of EP routers).

2) Energy consumption model of the hybrid nodes: To quantify the energy savings achieved by introducing optical bypass in an optical IPTV network, in this section we construct a power consumption model of the hybrid node of our solution.

Three factors affect the power consumption of a hybrid node. First, the power consumption of the router. Second, the power consumption of the OXC. Third, the power consumption of the OEO converters (transponders). Note that in this analysis we do not consider the power consumption of other optical equipment that is necessary in an optical network, such as the optical amplifiers, multiplexers and demultiplexers. Previous work [13], [12] has shown that switching equipment and transponders (OEO converters) are the main contributors for power consumption of optical IP networks (responsible for over 97% of total power consumption according to [13]), so we consider switching equipment and OEO converters only. Based on these three variables, we use the following model for the power consumption P of a hybrid node:

$$P = P_R + P_{OEO} + P_{OXC} \tag{4}$$

In Equation 4 P_R is the power consumption of the router, P_{OEO} is the power consumption of the OEO converters (transponders), and P_{OXC} is the power consumption of the optical cross connect. For $P_R + P_{OEO}$ we use the model represented by Equation 1. The power consumption of the OXC is given by Equation 5.

$$P_{OXC} = K_{op} P_{op} \tag{5}$$

In this equation, K_{op} is the number of input/output optical switch ports and P_{op} is the power per input/output switch port. We assume the OXC switching fabric to be based on micro-electro-mechanical systems (MEMS [22]). In a MEMS optical switch, a micro-mirror is used to reflect a light beam. The direction in which the light beam is reflected can be changed by rotating the mirror to different angles, allowing the input light to be connected to any output port. These MEMS have switching times of the order of milliseconds or hundreds of microseconds and for this reason can be used only for slow switching (i.e., circuit switching). For faster switching Semiconductor Optical Amplifiers (SOAs) could be used. But as MEMs consume less power [23], and as the OXC is not to be used for fast switching, MEMS are the option we consider here. We assume 3D-MEMS [24] in particular. The power per input/output switch port of the OXC corresponds to the power consumption for its continuous control, which is equal to 107 mW per input/output port. This value is based on the power consumption of the MEMS controller circuitry of an 80×80 3D-MEMS switch implementation, reported in [24]. We are therefore assuming power consumption is proportional to the number of active input/output ports[6]. The experimental figure and this assumption were considered in previous related work [23], [22] and are also in agreement with studies from other researchers [25], [10].

3) Results: We now analyse how the introduction of optical bypass techniques in the IPTV network translates into energy savings. We consider the same three scenarios as before: `150SD`, `700HD` and `3kUHD`. For the router model we also make the same assumptions. For the first scenario, we assume a router with 4 linecards with 4x1Gbps Ethernet ports each. For the other scenarios we just scale up the model by increasing the number and changing the type of linecards. This implies that

[6]If we assume an on/off behaviour, i.e., a switch consuming its 8.5 W of total power independently of the number of active ports, all results we present change by less than 1%. This stems from the fact that the OXC is the node component with the lowest power consumption by a good margin, in any case.

each wavelength can carry 1Gbps in the first two scenarios, but it scales to 40 Gbps in the third. Note that in this scheme two sets of wavelengths are needed: one for the traffic that optically bypasses the routers, and another for the rest. This is considered in the analysis to calculate the number of active OXC ports. The number of active OEO converters is equal to the number of active ports in the router.

In accordance with the results presented in Figure 5, we assume that 50% of the IPTV traffic optically bypasses the routers, 30% is sent to the router for electronic processing, and 20% of the TV channels are not distributed. Considering this, the power savings for all three scenarios (and the `3kUHD_ep` scenario) are presented in Figure 7. The dashed lines represent the results from using *selective pre-joining* only (the solid lines in Figure 6). The solid lines in the current figure represent the power savings using the optical bypass protocol proposed.

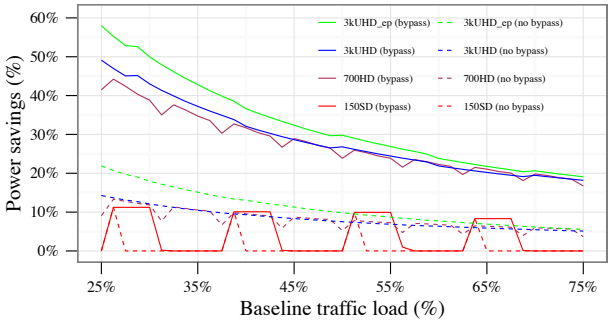


Fig. 7. Power savings achieved by optically bypassing popular TV channels in the network core

When compared with *selective pre-joining* [9], the introduction of optical bypass in the optical IPTV network core increases power savings substantially. At baseline traffic loads of around 30%, the power savings increase from 10% to over 40%, a significant four-fold increase. Considering EP routers, power consumption is halved. We conclude that the use of this technique is very effective in reducing power consumption, including in current IPTV service scenarios (such as `700HD`).

D. Discussion: on the value of electronics

We have just showed how optically switching *popular* IPTV traffic reduces power consumption significantly. How about optically switching *all* IPTV traffic? To answer, we invite the reader to Figure 8. This graph shows the result of optically switching all IPTV traffic in the network core (solid lines), against optically switching only the popular TV channels (dashed lines). As can be observed, by optically switching all IPTV traffic the power savings increase even further. Considering a baseline traffic load of 30%, in most scenarios an additional 15% power saving is achievable by all TV channels bypassing the routers.

So why not moving completely to optics in the future? In a scenario where all IPTV traffic is optically bypassed, to guarantee their availability for IPTV users all TV channels need to be distributed continuously in the network core. This is because OXCs allow slow switching only. The advantage of maintaining the electronic routing option is that, contrary to circuit-switched optical networks, with electronic routing it is

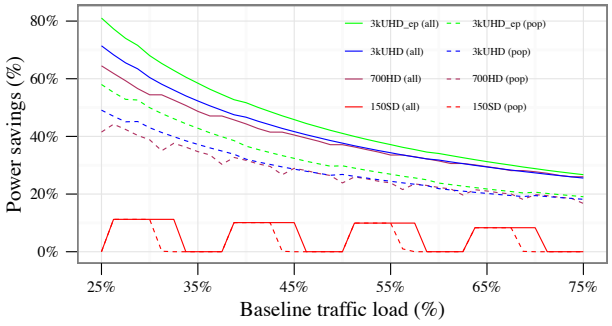


Fig. 8. Power savings achieved by optically bypassing all TV channels in the network core (compared to popular TV channels only)

possible not to distribute all TV channels. This added capability increases bandwidth efficiency. This can be seen in Table III, where we have included the bandwidth savings achieved by not distributing all TV channels. With the increased popularity of narrowcasting services and niche channels, the number of unpopular channels (as per our definition in this paper) may plausibly increase to the several hundreds or thousands in the near future. This trend offers an important argument for the maintenance of electronic routing as an option. A hybrid scheme as the one proposed therefore offers a compromise between *energy* and *resource* efficiency.

VI. Related work

IPTV measurement. With the recent deployment of IPTV networks a number of papers measuring and characterising IPTV traffic has been published [1], [3]. The analysis of real IPTV workloads led to a clearer understanding of how people watch TV and how this impacts the network. The findings from these studies and the analysis of our own dataset offered indications that led to the technique proposed in this paper.

Optics meet electronics. Optical switching techniques such as optical bypass – the technique we propose in this paper for IPTV – have been proposed as an interesting option to reduce the energy footprint of networks [26]. The problem, as explained before, is that due to its coarse granularity, bulk transport in optics can be bandwidth inefficient, especially for bursty traffic. With electronic switching the packets or flows can be processed at a much finer granularity. Smartly combining the strengths of optics and electronics has therefore been considered before [23]. For instance, Huang and Copeland [27] proposed a hybrid routing scheme that can preserve the benefits of optical bypass for large traffic flows and still provide multiplexing gain for small traffic flows. This technique is similar to the one we propose here for IPTV.

Green networking. Since the seminal paper by Gupta and Singh [28] the subject of *green networking* has received considerable attention. Several approaches have been considered to reduce energy consumption in networks, including performing resource consolidation by means of traffic engineering [28] or by putting components to sleep during periods of low traffic activity [29]. The literature in this subject is already substantial, so we refer the interested reader to a more detailed survey on green networking by Bianzino *et al.* [30].

Green IPTV. Recently, caching techniques for reducing energy consumption for time-shifted IPTV systems have been

proposed. By considering the particular properties of this type of traffic (which is in several aspects radically different from the live TV broadcasts we consider here), Nencioni *et al.* [7] proposed to cache content on local user storage thereby offloading traffic that would result from subsequent catch-up access. Osman *et al.* [8] also propose a caching strategy to store the most popular programs at nodes closer to the user, considering an IP-over-WDM network. The main differentiating factor of these works against ours is the fact that the IPTV services they consider are VoD or time-shifted broadcasts – not linear TV. Indeed, caching techniques are not suited for this type of service. In our previous work*et al.* [9], we have proposed a scheme that pre-joins only a selection of TV channels, instead of all, to save bandwidth and energy. The protocol we propose in this paper is based on a different technique that goes further in energy efficiency by considering the introduction of optical switching. As a consequence, we significantly improve the energy-efficiency of IPTV networks when compared to [9].

VII. Conclusions

In this paper we considered the introduction of energy-friendly optical technologies to reduce the energy consumption of IPTV distribution networks. We proposed an energy and resource-friendly protocol for the IPTV network core, blending electronic routing with all-optical switching. The main idea is to optically switch popular TV channels. This IPTV traffic bypasses the routers and therefore does not require any electronic processing (it is switched at the optical layer). The rest of the channels are sent to the routers for electronic processing (to be switched at the IP layer). By analysing a large dataset from an IPTV operator, we observed that with the proposed protocol it is possible to switch 50% of the IPTV traffic all-optically. The energy savings obtained from optically bypassing this traffic are substantial, reaching power savings of over 40% under normal load conditions. The scheme is also bandwidth efficient as channels without viewers are not distributed.

Acknowledgment

We would like to thank the anonymous reviewers for their feedback. This project has received funding from the European Union's FP7 research and Innovation programme under grant agreement No FP7-607109 (SEGRID) and from the Portuguese State through funds assigned by Fundação para a Ciência e a Tecnologia (FCT) to LaSIGE Research Unit, ref. UID/CEC/00408/2013.

References

[1] M. Cha et al. Watching television over an IP network. In *IMC*, Vouliagmeni, Greece, October 2008.

[2] Digital TV Research. Global IPTV Forecasts report. Technical report, Digital TV Research, 2014.

[3] T. Qiu, Z. Ge, S. Lee, J. Wang, J. Xu, and Q. Zhao. Modeling user activities in a large IPTV system. In *IMC*, Chicago, IL, November 2009.

[4] F. Legendre et al. Narrowcasting: an empirical performance evaluation study. In *CHANTS*, San Francisco, CA, September 2008.

[5] IPCC. Climate Change 2014: Synthesis Report. Technical report, The Climate Group, 2014.

[6] M. Webb. SMART 2020: Enabling the low carbon economy in the information age. Technical report, M. Webb, 2008.

[7] Gianfranco Nencioni et al. Understanding and decreasing the network footprint of catch-up tv. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, 2013.

[8] N.I. Osman et al. Energy-efficient future high-definition tv. *Lightwave Technology, Journal of*, 32(13), July 2014.

[9] F. M. V. Ramos et al. Reducing energy consumption in IPTV networks by selective pre-joining of channels. In *SIGCOMM workshop on green networking*, New Delhi, India, August 2010.

[10] J. Baliga et al. Energy consumption in optical IP networks. *Journal of Lightwave Technology*, 27(13):2391–2403, 2009.

[11] C. Fraleigh et al. Packet-level traffic measurements from the Sprint IP backbone. *IEEE Network*, 17(6):6–16, 2003.

[12] Y. Zhang et al. Energy efficiency in telecom optical networks. *IEEE Communications Surveys and Tutorials*, 12(4):441–458, 2010.

[13] G. Shen and R. S. Tucker. Energy-minimized design for IP over WDM networks. *IEEE/OSA Journal of Optical Communications and Networking*, 1(1):176–186, 2009.

[14] C.-H. Gan, P. Lin, and C.-M. Chen. A novel prebuffering scheme for IPTV service. *Computer Networks*, 53(11):1956–1966, 2009.

[15] C. Y. Lee, C. K. Hong, and K. Y. Lee. Reducing channel zapping time in IPTV based on user's channel selection behaviors. *IEEE Transactions on Broadcasting*, 56(3):321–330, 2010.

[16] T. Qiu et al. Modeling channel popularity dynamics in a large IPTV system. In *SIGMETRICS*, Seattle, WA, June 2009.

[17] P. Mahadevan et al. A power benchmarking framework for network devices. In *IFIP*, Buenos Aires, Argentina, June 2009.

[18] V. Sivaraman et al. Profiling per-packet and per-byte energy consumption in the NetFPGA Gigabit router. In *INFOCOM Workshop on Green Communications and Networking*, Shanghai, China, April 2011.

[19] Alcatel. Alcatel-Lucent WaveStar OLS 1.6T product specification. http://tinyurl.com/AlcatelWavestar. [Online; accessed 01-12-2015].

[20] W. Hou et al. Green multicast grooming based on optical bypass technology. *Optical Fiber Technology*, 17(2):111–119, 2011.

[21] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *IEEE Computer*, 40(12):33–37, 2007.

[22] M. Fiorani, M. Casoni, and S. Aleksic. Performance and power consumption analysis of a hybrid optical core node. *IEEE/OSA Journal of Optical Communications and Networking*, 3(6):502–513, 2011.

[23] S. Aleksic. Analysis of power consumption in future high-capacity network nodes. *IEEE/OSA Journal of Optical Communications and Networking*, 1(3):245–258, 2009.

[24] M. Yano, F. Yamagishi, and T. Tsuda. Optical MEMS for photonic switching-compact and stable optical crossconnect switches for simple, fast, and flexible wavelength applications in recent photonic networks. *IEEE Journal of Selected Topics in Quantum Electronics*, 11(2):383–394, 2005.

[25] R. S. Tucker. The role of optics and electronics in high-capacity routers. *Journal of Lightwave Technology*, 24(12):4655–4673, 2006.

[26] K. Hinton et al. The future Internet – an energy consumption perspective. In *OECC*, Hong Kong, China, July 2009.

[27] H. Huang and J. A. Copeland. Optical networks with hybrid routing. *IEEE Journal on Selected Areas in Communications*, 21(7):1063–1070, 2003.

[28] M. Gupta and S. Singh. Greening of the Internet. In *SIGCOMM*, Karlsruhe, Germany, August 2003.

[29] S. Nedevschi et al. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI*, San Francisco, CA, April 2008.

[30] A. P. Bianzino et al. A survey of green networking research. *IEEE Communications Surveys and Tutorials*, 14(1), 2012.

Pin it!
Improving Android Network Security At Runtime

Damjan Buhov*, Markus Huber†, Georg Merzdovnik*, and Edgar Weippl*
*SBA Research, Vienna, Austria
E-mail: {dbuhov, gmerzdovnik, eweippl}@sba-research.org
† St. Pölten University of Applied Sciences, St. Pölten, Austria
E-mail: markus.huber@fhstp.ac.at

***Abstract*—Smartphones are increasingly used worldwide and are now an essential tool for our everyday tasks. These tasks are supported by smartphone applications (apps) which commonly rely on network communication to provide a certain utility such as online banking. From a security and privacy point of view a properly secured (encrypted) communication channel is important in order to protect sensitive information against passive and active attacks. Previous research outlined that developers often fail to implement proper certificate validation in their custom SSL/TLS implementations and thus fail to secure the network communication. Previous research however proposed solutions for developers and not for the affected users. This global growth introduced drastic changes to the network utilization.**

In this paper we discuss this issue on the basis of Android apps. We analyzed over 50,000 Android apps, collected during two consecutive years, regarding the correct use of SSL/TLS protocols. Furthermore, we discuss the current situation. We propose dynamic certificate pinning, a device-based solution that overcomes the problem of broken SSL/TLS implementations in Android apps. To the best of our knowledge, we are the first to solve this problem by combining established techniques such as certificate pinning with dynamic instrumentation techniques to tackle one of the major security challenges in the network communication of smartphone applications.

I. Introduction

The increased use of smartphones also means more risks. People tend to part with the traditional desktop computers, even notebooks, and satisfy their requirements and needs with smartphones. This is possible, because all web services essential for personal and business use are rapidly transforming to meet the requirements of the mobile domain. The vast majority of those applications rely on network communication for data transfer, implying that applications that deal with sensitive user information are required to provide secure (*encrypted*) communication. Although there are multiple concepts that provide secure communication, Android applications most commonly use SSL (*Secure Sockets Layer*) and its successor TLS (*Transport Layer Security*) [16]. These two protocols are used to securely connect the client with a legitimate server. In Android, the security of SSL/TLS is in close interdependence with the client application. This means that the client (application) should employ proper verification methods for the server certificate, the hostname and deal with the SSL/TLS errors correctly. Hence, the verification logic is completely controlled by the application, or in other words the application developers are responsible for implementing the SSL/TLS certificate validation correctly. If such a validation scheme is not correctly employed, users face the risk of a *Man-in-the-Middle (MITM)* attack [28]. Such attacks can have significant implications especially in the financial sector, which increases the importance in the mobile domain. By default, the Android applications trust only certificates validated against the internal trust store, however, there are certain cases in which the developers need to implement custom validation of the certificates. Whether developers add custom implementation due to the need of applying additional protection measures such as *certificate pinning* or just because of problems they encounter with their self-signed certificates, the networking part of the application could easily become a vulnerable spot as a result of mistakes in these implementations. Since Android is the most dominant market share holder [15], such vulnerability will affect millions of users.

Recent research outlined that particular measures have to be taken in order to solve this issue. There have been many attempts, however, the situation has not changed. Tendulkar et al. [37] proposed that all the SSL/TLS configuration including pinned certificates etc. should be stated in the manifest file as a part of the application package. Back in 2012, Fahl et al. [26] examined the state of more than 13,000 applications regarding their SSL implementation. Using their script Mallodroid [12] they discovered that more than 1,000 applications from the data set were vulnerable to MITM attacks. All of the previous research efforts, however, propose instructions and solutions that are aimed towards the developers and not the affected users. This implies that even if there is a solution for the problem, developers might ignore this fact for various reasons. Another major drawback of using certificate pinning is the update interval of the applications, since it has been proven that from the time the update has been released by the vendor, until it reaches the users, makes the application totally unsuitable from a security point of view [31]. Thus, the goal of this paper is to detect and resolve this problem on a large scale and in real time, focusing on the affected users instead of the developers. For a single application or a very small set of applications it is possible to detect and fix this issue using static instrumentation. However, we aim for a dynamic device-based solution that overcomes the SSL/TLS issue and reaches

ISBN 978-3-901882-83-8

a large number of users. In summary, the contributions of this paper are:

- We analyzed the most popular 50,000 Android applications gathered over the period of two years and discovered that the SSL/TLS issue still exists and, more importantly, we found an increase of nearly 6% in terms of vulnerable applications from 2014 compared to 2013.
- We present a novel approach for patching Android applications during runtime that solves the SSL/TLS problem without any user interaction. Our solution employs established techniques such as *certificate pinning* combined with dynamic instrumentation techniques and provides a tool that can be installed on every Android device.
- To the best of our knowledge, we are the first to actually provide a solution that combines these techniques and directly affects the user, unlike most of the research effort that is aimed towards the developers.

The remainder of this paper is organized as follows. Section II provides the background and threat model of our work. Section III briefly describes our methodological approach. In Section IV we present a detailed explanation of the implementation of the proposed tool while Section V presents the results from the static analysis as well as evaluation of our tool. Section VI is reserved for discussion about the advantages, limitation and future work for the proposed solution. In Section VII we revise existing approaches that tackle this problem. Finally, Section VIII concludes the findings of our paper.

II. Background and Motivation

In this section we provide a brief overview on the security of Android with an emphasis on providing secure network communication. We furthermore discuss our threat model.

A. Android Security

Smartphones and tablets continue to replace the traditional desktop computer; Android has the biggest share of the overall smartphone market. The ongoing transition from desktop operating systems to mobile operating systems such as Android brings along a number of security improvements for the average user. Android relies on the concept of multilevel security [30] and, compared to traditional desktop operating systems, each Android application is executed in an isolated sandbox. In combination with Google's firm control over available applications with their Play Store[1], the impact of common security threats such as malware has been limited. While the overall security of Android outlines a number of security improvements, users still face security and privacy risks. These risks emerge from the ever growing amount of third-party applications. The development of third-party applications relies on the capability and knowledge of the developer. Their lack of knowledge or ignorance of security issues introduces grave implementation bugs in applications, which have an overall negative impact on the security of mobile users. Security of these applications is achieved through the use of the Android permission model and sandboxing which defines a particular memory space for the application to execute. This way the system ensures that only allowed resources will be available to the particular application. In most of the cases the requested resources by applications include permission to access the network/Internet. Hereby, the actual implementation within the application dictates the level of security for the network communications. In Android, SSL/TLS are the standards that enable secure communication and are widely used among the Android applications. Because of the fact that the security of SSL/TLS relies on certificates, proper implementation of the validation procedure for the server certificate is essential. By default, every Android device comes pre-shipped with 150+ root certificates.[2] These 150+ root certificates are used to ensure that applications can verify that they are communicating with legitimate servers. This basic network security model requires that application developers buy legitimate certificates from Certification Authorities (CAs). Nowadays, there are numerous Certificate Authorities that issue verified certificates, however, just 15 of them hold more than 95% of the market shares [2], [17]. There are cases in which the developers are using self-signed certificates. The use of self-signed certificates puts an obligation on the developer to ensure that proper security mechanisms are put in place to achieve the same level of security as with officially signed certificates.

B. SSL in Android

The implementation of SSL in Android is achieved through certain packages provided by the Android SDK [1]. Usually, developers make use of the *javax.net.*, java.net.*, android.net.*, java.security,*, org.apache.** modules which provide them with all the important interfaces such as *TrustManager* and *HostnameVerifier*. Furthermore, the TrustManager interface contains a method called *checkServerTrusted* through which the validation of the certificates is performed. Developers can choose whether to use the default configuration of the SSL/TLS or to implement their own custom version. Usually, the default configuration is used when the application is using trusted certificates, whereas custom implementation is required for any other case. In both cases, they must ensure that the validation of the certificates is properly implemented. This procedure can be described as follows:

- **Certificate verification** The server sends the chain of certificates to the application. At this point the application tries to validate the chain using the bottom-up approach, i.e. starting from the end certificate (also known as leaf certificate) and continue to the intermediate and root certificate. The validation of the certificates includes checks for the expiration date of the certificate and whether it is signed from its successor in the list or from a trusted root certificate. In this setup the last certificate is usually signed by one of the certificates that came with the device.

[1]https://play.google.com

[2]Location of the Root CAs: Settings → Security → Trusted credentials

If the validation succeeds, the connection is established; otherwise it is immediately terminated.

- **Hostname Verification** Another very important check is the hostname verification. Every certificate has its designated destination so the application has to check whether the certificate is issued for the desired destination. This information is usually found in the Common Name (CN) field or the subjectAltName. According to the newer standard [14], the subjectAltName should be checked first and if it exists, the CN field should not be checked at all.

Although this validation procedure works for certificates that are signed by some of the root certificates that are preshipped with Android, there are certain cases in which the developers need to implement their own logic. The most common reason behind this is the fact that most of the Android developers make use of self-signed certificates for various reasons, such as testing the product before official release, or simply because of financial reasons. When using this kind of certificates, developers are obligated to perform custom implementation of the validation procedure to make the application immune to the most common threat described in *II-D Threat Model*.

C. Certificate Pinning

Among all available advanced protection measures, *certificate pinning* [10], [11], [24], [29], [33] stands out as the most recommended one. With proper implementation, it reduces the risk of Man-in-the-Middle attacks to a minimum. This technique is the most common representative of the advanced concepts with respect to the custom use of SSL/TLS protocols that was previously mentioned. There are a lot of publicly available solutions that could be directly applied in order to secure the network communication and in particular Android applications. Certificate pinning works by bundling the server certificates with the application. Hereby, the application verifies the security of the network communication based on its included *Pins* (server certificates). Therefore, developers do not need to buy third-party certificates from CAs, and applications can, moreover, detect attacks in which trusted certificates are forged.

Since the application receives the whole chain of certificates from the server side, developers are left with the choice of which certificate to pin. Accordingly, pinning different certificates from the entire trust chain brings its own advantages as well as disadvantages:

- *Pinning the end certificate (leaf)* reduces the attack surface to a minimum since there are no certificates that are or could be signed from it; however, it is potentially subjected to a major drawback when it comes to change. These types of certificates are subjected to a change more often than the intermediate certificate, which implies that with every change of the leaf certificate the application has to be updated with the new pins, otherwise it will not be usable in terms of network connectivity.
- *Pinning the intermediate certificate* has a reasonably larger attack surface in comparison with the previous category, but it requires less updates since this certificate is not changed very often.
- *Pinning the root certificate* leaves the biggest attack surface compared to the previous two categories, however, in this case the update frequency is the lowest.

Furthermore, this technique could be applied both to the whole certificate as well as just to the public key of the certificate. Although it is the easier solution and in general it seems natural to pin the whole certificate, it is not the optimal one. The reason behind this is the fact that certificates can be reissued multiple times. This means that we can encounter multiple certificates with the same public key, but with different attributes, e.g. expiration date. Based on this assumption, we can conclude that it is more convenient to pin the public key of the certificate. Although the implementation of this approach is more difficult due to some extra steps regarding the key extraction, in the end, this approach significantly reduces the need for frequent updates of the application.

D. Threat Model

Our threat model regarding the network security of Android applications focuses on *Man-In-The-Middle (MITM)* attacks. MITM attacks describe a category of network-based attacks during which an adversary places himself between a client and a server. The adversary can then perform either passive or active attacks on the observed network traffic. Active attacks include hijacking active user session to perform malicious actions on behalf on the targeted users. Passive attacks include the collection of sensitive information such as account credentials or personal information. Proper use of certificates can prevent such attacks. If applications do not protect the communication between mobile devices and their backend servers, attackers can easily perform active/passive MITM attacks by e.g. monitoring users on public Wi-Fi hotspots. Our particular threat model focuses on applications that aim to protect their users with a secure communication channel (SSL/TLS), but fail to implement this protocol properly. In particular, our threat model accounts for the following network security challenges:

- **Broken Certificate Verification:** If an Android application uses certificates issued from one of those certificate authorities, which are shipped in with the device, it relies on the standard implementation of the SSL/TLS protocol and provides therefore basic security. The problem arises with the use of self-signed certificates. In this case, the developers should implement proper validation procedures in order to secure the connection. These custom implementations tend to leave the application insecure by implementing a broken certificate validation. The issue of a broken custom certificate validation remains a major problem of current Android applications and leaves applications as vulnerable to MITM attacks as if no encryption was used at all.
- **Compromised Certification Authorities:** Even in cases in which developers rely on certificates issued by trusted

certification authorities and the default verification mechanisms of Android, powerful adversaries might still perform MITM attacks. The Diginotar case [9] clearly showed the risk of trusted certificate authorities (CAs) being compromised by adversaries. If an attacker is able to compromise one of the 150 CAs trusted by Android, he can perform MITM attacks on applications with the standard SSL/TLS protection.

To account for the attack vectors of our threat model, we propose a solution to dynamically pin application certificates. Hereby, we rely on the *Trust on First Use (TOFU)* principle. Since we do not have any previous information about the certificate that is going to be pinned, we use this approach to get the first certificate that the application will receive during the establishing of the SSL/TLS connection. Therefore, our threat model assumes that the first connection between a given mobile application and their corresponding servers is not compromised. Based on our threat model we aim to overcome the following challenges regarding the network security of Android applications:

- **Dynamically upgrade applications to use certificate pinning.** If the implementation of the pinning is not correct, the user faces grave security and privacy consequences. Users become an easy target for adversaries to steal sensitive information such as banking credentials, social security numbers, etc. We attribute the lack of proper SSL/TLS implementations to a knowledge gap of the developers. Previous research showed e.g. that the vast majority of developers are not familiar with the concept of certificate pinning [31]. Most of the time they are guided by random forum posts and discussions on popular online forums such as stackoverflow.[3] It so happens that a number of posts related to the use of SSL/TLS on Android actually advise developers to handle the SSL/TLS errors by accepting all certificates. By doing this, they actually remove any security on the network level, even the (secure) default setting, because most of the posted solutions suggest to use custom implementations of the *TrustManager* [18] which overrides the default one. Therefore, we aim at overcoming this knowledge gap by proposing a solution that focuses on the affected users instead of developers.
- **Providing the users with detailed information for every certificate change.** In order not to significantly lower the usability of the Android applications by immediately terminating the connection when a certificate change occurs, we have provided the users with a notification containing detailed description for the change that just occurred. At this point, users are left with the possibility to accept this change and continue to use the application, or to reject it, which would imply that the connection will be terminated immediately.

[3]http://stackoverflow.com

III. Methodology

A. Number of vulnerable applications

Generally speaking, Android can be divided into two main parts: The first part is the Android operating system, and the second one are the Android applications. We do not focus on the overall security of the Android operating system, but rather on applications and the not-so-obvious threats presented by them. Nowadays, there are more than 1.5 million available applications in the official Android market place [13]. Taking in consideration the popularity of the Android OS, we firstly analyzed the top 50,000 applications from all categories over two consecutive years. This allowed us to determine to what extend SSL/TLS issues are present in these two sets of applications. These analyses are focusing directly on the correct implementation of the HTTPS protocol, namely the *TrustManager*. The cases in which the applications are using pure HTTP are not taken into consideration and are immediately classified as applications that do not have any issues. To perform our analysis, we rely on the *Mallodroid* tool which explicitly targets the implementation of the *TrustManager*. In detail, we performed an indicative experiment in order to determine if the SSL/TLS errors still exist. Detection of other network flaws or tracking the evolution of particular applications is out of the scope of this work.

The applications were already crawled by *Playdrone* [39] and are publicly available at archive.org. We selected the top 25,000 applications from late 2013 and resp. from late 2014. According to [39], crawled applications originate from different categories. Finally, this experiment should help to understand if SSL/TLS implementation errors continue to put user data at risk or if the situation improved.

B. Fixing a broken trust manager

Since Android applications are packages stored across servers (market place), we do not have access to nor are we permitted to perform any changes to them. The situation is however different when the applications are downloaded and installed on a certain device, since the user has already agreed on all of the previously presented terms, known as Android permissions. In general, our approach does not interact or change the code of the application as static instrumentation would. We tackle the problem of broken SSL/TLS implementations dynamically, leaving the apk[4] intact. We are able to achieve this by leveraging the functionality of the Cydia Substrate framework [35]. We chose to use Cydia because of two reasons:

- It is available for other smartphone operating systems which increases the chances of a widespread use of our approach.
- It is the only framework that supports the Android permission model. This means that even though it requires ROOT access, its use must be explicitly specified in the Android manifest file.

[4]Android application package

The overall goal of our approach is to evaluate a proof-of-concept solution of our dynamic approach to fix the SSL/TLS issue from the users' perspective.

IV. Design

Our proof-of-concept implementation is based on the dynamic instrumentation of mobile applications [19], [35]. These dynamic instrumentation frameworks are especially popular among the users of custom ROMs such as *CyanogenMod* [4]. Today, the most common use of these frameworks consists in the creation of customized widgets and other GUI elements. The only requirement that has to be met for proper use of these frameworks is the available ROOT access to the device. This unleashes the full power of the frameworks, expressed through the possibility of interception, hooking and modification of functions, system calls and class loading, interpreted both through Java and native code. The fact that they operate at runtime enables us to intercept and modify all networking calls. Furthermore, the wide use of these frameworks in communities that rely on custom ROMs renders our tool as a promising candidate for securing the network communication. Recent statistics [5] [3] show that there are currently more than 50 million people using CyanogenMod on their smartphones. The vast majority of those users are strongly focused on privacy and security enhancements, especially after the Edward Snowden revelations.

In our previous research [22] we identified the most suitable candidates to achieve our goal. We thus decided to use the Cydia Substrate framework [6], [35] for our dynamic approach. We chose this framework because it is the only framework that is available for the two leading smartphone operating systems – Android and iOS. In contrast to previous research – which is mostly focused on the developers – our focus group are the users. Furthermore, our solution presents itself as an OS extension, so it could be easily included as a factory feature. Cydia Substrate (formerly known as Mobile Substrate) serves as a base for the development of particular tools/modules. It provides a set of different APIs which can be adapted according to the specific needs. In general, these APIs make it possible to get a reference to a particular class of the applications with the *MSJavaHookClassLoad* and then search inside that class for the desired method or function that should be hooked with the *MSJavaHookMethod*. The last step would be to replace the code of the hooked function or method with our custom implementation. This procedure could be easily applied to all generic calls, however, in some cases static instrumentation of the code might be needed in order to detect the hooking point. Our implementation consists of two classes, one for the implementation of the hooking functionality, and the other for the pinning trust manager. Since SSL/TLS implementations in Android apps are dependent on certain API calls that are provided in the tutorial itself [20], we do not have to perform any static instrumentation to distinguish these calls. Instead, we can directly interact with those API calls. Using the previously mentioned *MSJavaHookClassLoad* function, we wait for the *javax.net.ssl.TrustManagerFactory* to load and search for the *getTrustManagers* method to be hooked. Upon hooking, the current implementation of the *getTrustManagers* is overridden by our custom implementation. This set of instructions directly influences the current implementation by substituting it with our version. Furthermore, additional changes have to be made for the application to work properly. We then override the *setSSLSocketFactory* method upon loading of the *javax.net.ssl.HttpsURLConnection* and setting it to use our implementation of the trust manager. Last but not least, we override the *init* method from the *javax.net.ssl.SSLContext* class to use our *TrustManager*. This way we ensure that every established connection will be pinned. Although there are different ways to verify the hostname, we are using strict verification. This means that every pin is associated with its designated host. By using the *TOFU* principle, we pin every connection upon the first encounter. Therefore, every pin is associated with the designated host, and when the connection is trying to be established later on, the hostname along with the pin for that particular connection is checked. Whenever there is a mismatch in any of the fields, whether it would be the hostname or the pin itself, a notification will alert the user immediately. Instead of terminating the connection if the certificate changes, which can happen without any malicious intent, we enable the user to decide whether to approve the change and pin the new certificate or to reject the change and terminate the connection. The users are included in this process, because if we directly terminate the connection, it will render the applications unusable.

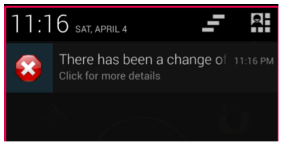
V. Results

We conducted a static analysis of 50,000 Android applications. The static analysis is solely focused on the *TrustManager* implementation within the applications. The applications dated from two consecutive years. One set of top 25,000 applications was crawled in late 2013 and the second set in late 2014. For our analysis we used the *Mallodroid* script [12] and the results are categorized according to the following criteria: *Broken TrustManager*, *Possibly Broken TrustManager*, *Broken hostnameVerifier*, *Possibly Broken hostnameVerifier*, *Broken SSLError Handling* and *Possibly Broken SSLError Handling*. The results confirmed that the applications rely more and more on network communication. The results are presented in Table I and Table II.

	Trust Manager	Hostname Verifier	SSL Error
Broken	17%	7%	0.08%
Possibly Broken	6%	1%	15%
No issues	54%		

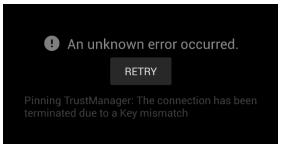
TABLE I: Classification of applications that contain Broken and Possibly broken TrustManager, Hostname Verifier and SSL Errors for the set of 25,000 applications from late 2013

It is evident that the situation is just getting worse. The top 25,000 applications from late 2013 contained 3,834 applications or nearly 17% that had no implementation or a broken custom implementation of the validation procedure.

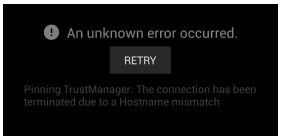


(a) Screenshot from the notification in the Notification Bar

(b) Screenshot from the body of the notification showing the change of the certificate that occurred



(c) Example of terminated network connection due to key mismatch



(d) Example of terminated network connection due to hostname mismatch

Fig. 1: Screenshots of the notification and the error messages produced by a mismatch of the public keys and the hostnames – simulation of *MITM* attack

	Trust Manager	Hostname Verifier	SSL Error
Broken	23%	13%	0.05%
Possibly Broken	10%	4%	29%
No issues	21%		

TABLE II: Classification of applications that contain Broken and Possibly broken TrustManager, Hostname Verifier and SSL Errors for the set of 25,000 applications from late 2014

```
public TrustManager(){
  return;
  }
public void
    checkServerTrusted(java.security.cert.
X509Certificate[]s1, String s2){
  return;
  }
```

Listing 1: Example code for Broken TrustManager that would accept all certificates

Usually, such a set of applications also contains the other two categories, i.e. broken hostname and no handling of SSL/TLS errors. Rather surprisingly for us, the applications from 2014 turned out to have more broken SSL/TLS implementations. As seen in Table II, 23% resp. 4,804 applications have a broken SSL/TLS implementation or are set to accept every certificate that is presented to them. This increase of nearly 6% is a clear indication that the problem still exists among the Android applications that make use of the SSL/TLS protocols for securing their network communication. Although it is evident that the problem still exists, moreover we notice an increase in the applications that contain broken SSL/TLS implementation from the set of 2014, it could be an indication that the awareness towards this issue has finally raised. This is because of the functionality of the *Mallodroid* script which targets just the applications that are using SSL/TLS, specifically the *TrustManager* implementation, while the applications that use just HTTP are immediately classified as applications that contain no issues. Due to the design of this script, additional network flaws are also not registered. This indicates that all of the applications that use just HTTP or do not use Internet are classified under the *No Issues* category. Moreover, this increase could be classified as mixture of fast adoption of the concepts providing additional SSL/TLS security combined with the lack of knowledge with regard to the actual implementation of these concepts. The lack of a centralized body (such as Google Play is for testing the applications regarding all additional threats) that is capable of testing the actual implementation of the networking part of the application could easily introduce such increase in the results, since Google itself has a quite open approach towards the process of becoming a developer without assessing their actual qualifications. This implies that even with increased awareness, there is no guarantee that the implementation will be correct. Finally, we handpicked a very small set of already identified applications with broken SSL/TLS implementation for further analysis. In this set of apps, we encountered classes named *FakeTrustManager*, *AcceptAllTrust* etc. and found copied chunks of code directly from the forums that advice users to trust all certificates in order to solve the SSL/TLS errors in their applications. An example of a detected broken implementation is outlined in Listing 1.

After having presented the design of our tool, we assess its effectiveness. In order to be able to test our tool, we did a setup that includes Android devices with root access, Cydia Substrate installed and our tool, which in turn was installed

Fig. 2: Screenshot from the file that contains the public key pins and the hostnames

as an extension to the framework. As specified by the Cydia Substrate framework, after installation of a new module the phone has to be rebooted. From the point when the phone is booted, the public key of every certificate for every connection is automatically pinned. Figure 2 shows the pinned public keys.

Our implementation of the pinning is based on the suggestions from the OWASP guide [33]. Although it might require frequent updates, we decided to pin the end certificate in order to reduce the attack surface as much as possible. By pinning the public key of the certificate, we introduced the user with a little bit more flexibility compared to pinning the whole certificate. This means that the applications that use our approach will eliminate the need for an update when the certificate is reissued, since the public key will be the same.

We manually verified our solution against a number of applications that had a completely broken use of SSL/TLS protocols. Upon applying our solution, all of the tested applications were functioning as expected and their public keys were successfully pinned. In order to simulate a MITM attack, we manually changed the pinned keys in the file. This means that next time when the application tries to establish a network connection, our solution should send an alert to the user. After successfully changing the key and reopening the application, we received an alert that the key had been changed (shown in Figure 1a and Figure 1b). Here the user can decide whether he accepts this change and pin the new certificate or just reject it. If the user pins the new key, the application will continue to work normally, since in our case the certificate that is received is the valid one from the application, because we performed the change directly in the file that contain the pins. In general, if the user rejects the change of the certificate, the connection will be immediately terminated. An example of this case can be seen in Figure 1c and Figure 1d. Furthermore, the tests performed against the set of applications showed no decrease in the overall performance of the applications.

VI. Discussion

Our approach improves the network security of broken/default SSL implementations in Android applications by dynamically pinning server certificates. Hereby, we directly improve the security of affected users instead of focusing on the developers of the applications. We also managed to shift the pinning strategy from an application-based approach to a broader device-based solution. Furthermore, we do not have any limitations regarding the number of applications, because our approach works with all Android applications that are using network communication.

A. Security Challenges

Besides all the benefits presented throughout this paper, our current proof-of-concept implementation has open challenges. Since we are relying on the *TOFU* principle, our tool works only if the first connection is benign. Although there is still a risk that the first connection might be malicious, it is definitely significantly lower in comparison to the risk of MITM attacks on a totally insecure application. Furthermore, our approach could be easily adapted to overcome the TOFU issue in future work. As proposed by Wendlandt [41], we plan to implement a third-party notary service to ensure that applications use the correct pin, even when the first connection has already been tampered with. This solution enables the possibility to provide pins in advance and, more importantly, offers a distributed infrastructure to detect MITM attacks. By implementing such an extension, the application will directly receive the pins while the risk of a first malicious connection is eliminated, because the provided pins are verified against our third-party notary service. Furthermore, it also overcomes the limitation presented by the certificate updates, because new certificates will be verified and the user will be accordingly informed whether the certificates are malicious. An other alternative approach to overcome the limitation presented with the use of the *TOFU* approach is to use the DNS-based Authentication of Named Entities *(DANE)* [7]. This alternative would, however, also require the support of DANE by applications developers.

Our approach requires root privileges to work. Therefor, an attacker would also need root privileges to subvert our security improvements by e.g. tampering with the pin storage of our tool. In scenarios where attackers are able to gain root access to a device, they can already access any information stored on the device directly and would likely not focus on subverting our protection mechanisms. It is also important to state that our approach is not directly exposing the system to any additional risks apart from the ones already presented with the use of the root mode in Android operating system. Finally, to address the cases in which applications already have a correct implementation of the SSL/TLS protocols, we will introduce application whitelisting. This means that applications with a correct implementation of SSL/TLS could be excluded and not obligated to use our implementation of the *TrustManager*.

B. Usability Challenges

Our proof-of-concept implementation requires user interaction and can thus be compared with the current implementation of common web browser warnings, during which users are explicitly asked to decide whether to proceed or terminate the connection to the desired web service. Due to the specific nature of real-world usability testing of our module, a large-scale long-term usability test would provide additional insights regarding the user acceptance of our approach. In addition, an adaption of the previously mentioned notary-based or DANE-based pinning approach would also minimize the requirement for users to decide if a given certificate is valid or not. Our work touches upon another important issue related to secure network communication that is still present in the Android OS: the lack of visual indicators. Unlike browsers, where the user is notified with the lock in the address bar when HTTPS is used, in Android there is no way for the user to distinguish whether the user is using a secure channel to transmit sensitive data or not. Our approach could therefor be used to inform the user when secure network communication is used for a specific app. Finally, we plan to make our approach even more accessible by porting our proof-of-concept implementation to iOS.

VII. Related Work

Taking into consideration the market share of Android, it is obvious that any vulnerability would affect a large number of users. From the start, researchers put a lot of effort in discovering bugs and proposing solutions. It is the same with the networking part of the applications that are currently on the official market. Trummer et al. [38] and Onwuzurike et al. [32] recently showed that some of the most popular applications currently available are still vulnerable to MITM attacks. This underlines that the problem still exists and according to [27], lack of knowledge is one of the reasons for this issue. While not being able to directly influence this matter, researchers turn to proposing tools for static analysis that could help developers and researchers to detect broken SSL/TLS implementations. Sounthiraraj et al. [36] proposed SMV-Hunter, a tool that combines static and dynamic analysis to detect incorrect use of SSL/TLS protocols. Zuo et al. [42] presented a hybrid approach to discover these vulnerabilities. They analyzed 13,820 applications and found out that 1,360 are potentially vulnerable. The drawback of all those solutions is that they are focused on the developers. In contrary, our approach is aiming at a more scalable solution: we developed a module for dynamic certificate pinning which scales the common application-based approach to a broader device-based and user-focused solution. Instead of limiting the certificate pinning to just one app, our module is able to implement this approach for every single application installed on the device. Furthermore, all of the past research is performed over a fixed set of applications, whereas our focus is put on the users, i.e. without having a limitation for the application set. This means that we are able to apply our solution to any application that is utilizing the network. This way we directly influence the user instead of the developers.

Network security is just a part of the whole Android security model, therefore we refer the interested user to Enck et al. [25] for a detailed explanation of the Android operating system as well as its overall security concept. Currently a number of researchers focus on discovering applicable attack vectors for the Android operating system. Bugiel et al. [21] and Davi et al. [23] present privilege escalation attack vectors that underline weaknesses in the Android operating system. Finally, the Android permission has received considerable attention from the research community. Information regarding the evolution and effectiveness of the permission system as well as detailed studies regarding over-privileged applications can be found in [34], [40].

VIII. Conclusion

In this paper we discuss a major security and privacy issue of Android applications: weak protection of the network communication between devices and backend servers. To this end we performed a static analysis on 50,000 Android applications gathered over a period of two years. Our analysis showed that the broken implementation of SSL/TLS communication remains a serious issue for popular Android applications. Our analysis suggests that this issue did not improve over time. We furthermore present a novel approach to overcome the issue of broken SSL/TLS use in Android applications. Hereby, we proposed a tool that provides dynamic pinning of certificates during runtime. To the best of our knowledge, we are the first to tackle this major security challenge from the users' perspective. Our approach is based on a popular dynamic instrumentation framework which is available for the great majority of Android devices and, thus, makes our proposed implementation a suitable candidate for future custom ROMs. Therefore, we made the source code of our proof-of-concept implementation publicly available [8] to spur adaption of our approach in popular custom Android ROMs such as CyanogenMod.

Acknowledgment

This research was funded by COMET K1, FFG – Austrian Research Promotion Agency. Moreover, this work has been carried out within the scope of "u'smile", the Josef Ressel Center for User-Friendly Secure Mobile Environments, funded by the Christian Doppler Gesellschaft, A1 Telekom Austria AG, Drei-Banken-EDV GmbH, LG Nexera Business Solutions AG, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH.

References

[1] Android sdk. [Online]. Available: http://developer.android.com/sdk/index.html
[2] Certificate authority. [Online]. Available: https://en.wikipedia.org/wiki/Certificate_authority
[3] Cyanogen usage statistics. [Online]. Available: http://www.androidcentral.com/cyanogen-now-has-more-users-windows-mobile-and-blackberry-combined
[4] Cyanogenmod. [Online]. Available: http://www.cyanogenmod.org
[5] Cyanogenmod statistics. [Online]. Available: http://www.digitaltrends.com/mobile/does-cyanogen-really-have-more-users-than-windows-mobile-and-blackberry-combined/

[6] Cydia substrate apk. [Online]. Available: https://play.google.com/store/apps/details?id=com.saurik.substrate

[7] Dns-based authentication of named entities (dane). [Online]. Available: https://tools.ietf.org/html/rfc6698

[8] "Dynamic pinning solution." [Online]. Available: https://github.com/dbuhov/pinningTrustManager

[9] Final report on diginotar hack shows total compromise of ca servers. [Online]. Available: https://threatpost.com/final-report-diginotar-hack-shows-total-compromise-ca-servers-103112/77170/

[10] M. marlinspike. tack - trust assertions for certificate keys. [Online]. Available: http://tack.io/draft.html

[11] M. marlinspike. your app shouldn't suffer ssl's problems. [Online]. Available: http://www.thoughtcrime.org/blog/authenticity-is-broken-in-ssl-but-your-app-ha/

[12] Mallodroid script - https://github.com/sfahl/mallodroid. [Online]. Available: https://github.com/sfahl/mallodroid

[13] Number of available applications in the google play store. [Online]. Available: http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[14] Representation and verification of domain-based application service identity within internet public key infrastructure using x.509 (pkix) certificates in the context of transport layer security (tls). [Online]. Available: https://tools.ietf.org/html/rfc6125

[15] Smartphone os statistics 2015. [Online]. Available: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qpsp=2015&qpnp=1&qptimeframe=Y

[16] The transport layer security (tls) protocol version 1.2. [Online]. Available: https://tools.ietf.org/html/rfc5246

[17] Usage of ssl certificate authorities. [Online]. Available: http://w3techs.com/technologies/overview/ssl_certificate/all

[18] X509trustmanager. [Online]. Available: http://developer.android.com/reference/javax/net/ssl/X509TrustManager.html

[19] Xposed framework. [Online]. Available: http://repo.xposed.info

[20] Android. Security with https and ssl. [Online]. Available: http://developer.android.com/training/articles/security-ssl.html

[21] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A. Sadeghi, and B. Shastry, "Towards taming privilege-escalation attacks on android," in *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. [Online]. Available: http://www.internetsociety.org/towards-taming-privilege-escalation-attacks-android

[22] D. Buhov, M. Huber, G. Merzdovnik, E. Weippl, and V. Dimitrova, "Network security challenges in android applications," in *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, Aug 2015, pp. 327–332.

[23] L. Davi, A. Dmitrienko, A.-R. Sadeghi, and M. Winandy, "Privilege escalation attacks on android," in *Proceedings of the 13th International Conference on Information Security*, ser. ISC'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 346–360. [Online]. Available: http://dl.acm.org/citation.cfm?id=1949317.1949356

[24] N. Elenkov. Certificate pinning in android 4.2. [Online]. Available: https://github.com/nelenkov/cert-pinner

[25] W. Enck, M. Ongtang, and P. McDaniel, "Understanding android security," *Security Privacy, IEEE*, vol. 7, no. 1, pp. 50–57, Jan 2009.

[26] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith, "Why eve and mallory love android: An analysis of android ssl (in)security," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 50–61. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382205

[27] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith, "Rethinking ssl development in an appified world," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 49–60. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516655

[28] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous code in the world: Validating ssl certificates in non-browser software," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 38–49. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382204

[29] M. Marlinspike. Android pinning. [Online]. Available: https://github.com/moxie0/AndroidPinning

[30] J.-S. Oh, M.-W. Park, and T.-M. Chung, *The Multi-level Security for the Android OS*, B. Murgante, S. Misra, A. Rocha, C. Torre, J. Rocha, M. Falcão, D. Taniar, B. Apduhan, and O. Gervasi, Eds. Springer International Publishing, 2014, vol. 8582.

[31] M. Oltrogge, Y. Acar, S. Dechand, M. Smith, and S. Fahl, "To pin or not to pin—helping app developers bullet proof their tls connections," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015, pp. 239–254. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/oltrogge

[32] L. Onwuzurike and E. De Cristofaro, "Danger is my middle name: Experimenting with ssl vulnerabilities in android apps," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, ser. WiSec '15. New York, NY, USA: ACM, 2015, pp. 15:1–15:6. [Online]. Available: http://doi.acm.org/10.1145/2766498.2766522

[33] OWASP. Certificate and public key pinning. [Online]. Available: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

[34] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Android permissions: A perspective combining risks and benefits," in *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '12. New York, NY, USA: ACM, 2012, pp. 13–22. [Online]. Available: http://doi.acm.org/10.1145/2295136.2295141

[35] Saurik. Cydia substrate. [Online]. Available: http://www.cydiasubstrate.com

[36] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, and L. Khan, "Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps," in *Proceedings of the 19th Network and Distributed System Security Symposium*, 2014.

[37] V. Tendulkar and W. Enck, "An application package configuration approach to mitigating android SSL vulnerabilities," *CoRR*, vol. abs/1410.7745, 2014. [Online]. Available: http://arxiv.org/abs/1410.7745

[38] T. Trummer and T. Dalvi. The savage curtain: Mobile ssl failures, black hat - https://www.blackhat.com/docs/ldn-15/materials/london-15-trummer-dalvi-the-savage-curtain-mobile-ssl-failures-wp.pdf.

[39] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, pp. 221–233, Jun. 2014. [Online]. Available: http://doi.acm.org/10.1145/2637364.2592003

[40] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Permission evolution in the android ecosystem," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/2420950.2420956

[41] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving ssh-style host authentication with multi-path probing," in *USENIX 2008 Annual Technical Conference*, ser. ATC'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 321–334. [Online]. Available: http://dl.acm.org/citation.cfm?id=1404014.1404041

[42] C. Zuo, J. Wu, and S. Guo, "Automatically detecting ssl error-handling vulnerabilities in hybrid mobile web apps," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. New York, NY, USA: ACM, 2015, pp. 591–596. [Online]. Available: http://doi.acm.org/10.1145/2714576.2714583

End-to-end transparent transport-layer security for Internet-integrated mobile sensing devices

Jorge Granjal
DEI/CISUC, University of Coimbra
Polo 2, Pinhal de Marrocos, 3030-290
Coimbra, Portugal
jgranjal@dei.uc.pt

Edmundo Monteiro
DEI/CISUC, University of Coimbra
Polo 2, Pinhal de Marrocos, 3030-290
Coimbra, Portugal
edmundo@dei.uc.pt

***Abstract*—End-to-end communications with Internet-integrated sensing devices will contribute to the enabling of many of the envisioned IoT applications. Communication technologies with this purpose are currently being designed based on the 6LoWPAN adaptation layer, and of particular interest is CoAP (Constrained Application Protocol). The support of security in end-to-end CoAP communications with mobile Internet-integrated sensing devices is currently a challenge, in particular because of the high cost of performing ECC computations in constrained wireless sensing devices. Other important aspects to consider are the incompatibility of end-to-end security with CoAP proxies and the usage of mobile sensing devices.**

The mechanisms described in the article offer a practical solution to the previous challenges. We propose a transparently mediated DTLS handshake with mutual authentication and mobility support, with the goal of releasing constrained sensing devices from the burden of having to support costly ECC computations. We employ pre-shared key authentication in sensing devices, together with an authentication protocol for mutual authentication and confidentiality in the WSN side of end-to-end communications. From our experimental evaluation on the impact of the proposed mechanisms on the energy and computational effort required from sensing devices, we are able to verify that the proposed approach is viable in various usage scenarios. Overall, the proposed approach works transparently for the applications running on the Internet clients and sensor devices. It is our goal that, with the proposed mechanisms, distributed IoT applications may benefit from pervasive and transparent end-to-end security, irrespective of the static or mobile nature of the sensing devices employed. Ours is, as far as our knowledge goes, the first proposal with such goals.

Keywords—End-to-end transport-layer security, DTLS mobility, delegated public-key authentication, ECC, 6LoWPAN, DTLS, CoAP

I. INTRODUCTION

Most of the applications envisioned for the Internet of Things (IoT) are critical in respect to security, either of its users or of the data stored and transferred between devices. On the other hand, researchers know very well that the constraints in resources of sensing devices difficult the employment of traditional security approaches and mechanisms. This remains true if we focus on end-to-end communications with Internet-enabled devices employing 6LoWPAN-based communication technologies. In fact, technologies such as 6LoWPAN [1-3] and CoAP [4,5] are being designed precisely to enable the usage of constrained sensing devices as full Internet citizens, but challenges remain in what concerns security, in particular for end-to-end communications with such devices and when such communications are with devices that by nature are mobile.

In this article we start by proposing a model for the interconnection of low-energy wireless communication domains with the Internet, and in the context of this model we propose a set of mechanisms designed with the purpose of supporting end-to-end security with mobile sensing devices. The proposed mechanisms allow us to offer practical and effective solutions to three aspects currently representing research challenges in the area: the high cost of end-to-end transport-layer security for constrained wireless sensing devices, the incompatibility of end-to-end security with the usage of proxies, and the lack of mechanisms to abstract end-to-end communications and security from the movement of sensing devices. Our proposals address the previous challenges, while guaranteeing total compatibility with the mechanisms already adopted.

The article is structured as follows. In the next Section we discuss our motivations, and Section III presents the proposed integrated model for end-to-end security with mobile devices. The mechanisms proposed in the context of this model are discussed in Section IV and experimentally evaluated in Section V. Section VI discusses related work and Section VII finally concludes the article.

II. MOTIVATION

Contrary to the perception of researchers a few years ago, the emergence of 6LoWPAN-based communication technologies [1-3] is enabling Internet communications with constrained sensing platforms. Distributed IoT applications may employ CoAP [4,5] at the application-layer, in order to retrieve resources from sensing devices, or for autonomous communications between WSN and Internet devices. CoAP is being designed to enable application-layer RESTful communications with such sensing platforms, and it promises to be a cornerstone for the support of future IoT applications. The addressing of security in

ISBN 978-3-901882-83-8 © 2016 IFIP

the context of CoAP is thus of major importance although, as we discuss next, various issues still complicate effective security.

The current CoAP specification adopts DTLS (Datagram Transport Layer Security) [6] at the transport-layer security with the goal of transparently securing CoAP communications at the application-layer. DTLS provides security that, by nature, is end-to-end, but in reality conflicts with another functionality designed in CoAP: the usage of proxies to assist communications between the Internet and WSN communication domains. Another aspect currently motivating research efforts is that DTLS, as adopted for CoAP, requires the usage of public-key authentication using ECC (Elliptic Curve Cryptography) for authentication and key agreement. ECC is well know to be too resource demanding in constrained sensing devices, further complicating the adoption of DTLS in practical applications. Another aspect is that many IoT applications may employ devices that by nature move from one WSN domain to another, even if between WSN domains under the same administrative control. Thus, mechanisms are also required to support inter-WSN mobility in the context of end-to-end communications and security, as we address in this article. We address the previous aspects in an integrated fashion, proposing a coherent solution to address the limitations of CoAP security.

As already discussed, DTLS is currently mandatory for CoAP, the same applying to the support of ECC public-key cryptography. It is well accepted that ECC is still too costly for sensing platforms such as the TelosB [7], and this aspect currently motivates various research proposals, as we discuss in Section VI. Our proposal consists in the offloading of costly computations related with the handshake to a more capable device, at the same time guaranteeing total transparency from the point of view of the communicating entities and applications. In particular, we extend our previous proposal on DTLS authentication with mediation [8] to include support for mobile sensing devices. The costliest phase of DTLS is the initial authentication and key agreement handshake, and our proposal not only supports the offloading of ECC computations to a router, but also works side-by-side with our mobility model, allowing for inter-WSN movement of CoAP sensing devices.

As previously referred, DTLS as currently considered for CoAP conflicts directly with the usage of CoAP proxies, either in reverse or forward mode. This is in fact a concern, as CoAP proxies are useful and a necessity in many scenarios. By intercepting and mediating the DTLS handshake our model offers an effective solution to the support of CoAP proxies, since the same entity can support all functionalities.

Regarding the mobility of sensing devices, our goal is to propose mechanisms that can abstract IoT applications, and also end-to-end communications and security, from the actual position of a device inside a WSN administrative domain. A device may roam between different WSN domains inside a given administrative domain (e.g. in medical applications, where patients in an hospital carry a sensing platform and may move between different networks, or in industrial monitoring and control applications), while applications still are able to establish end-to-end communication and secure sessions with the device. We note that mobility will be in fact an important requirement of many IoT applications, for example, sensors may be attached to moving machinery in a factory or building, or to a vehicle moving around in a plant, or even used for biometric purposes and attached to persons. Our proposal considers that mobility is a reality, and also that end-to-end communications between Internet hosts and CoAP devices must be maintained for sensing devices moving in the same administrative domain. Thus, even with security such devices are able to keep serving CoAP requests from Internet hosts, in the context of the application.

Overall, the contributions in this article belong in the context communication and security technologies based on 6LoWPAN, that are already contributing to the formation of an IoT communications stack as analysed in [9]. More precisely, our aim is to contribute to security in the context of this stack, and offer what we believe are effective solutions to the problems previously identified.

III. An Integration Model for End-to-End Security with Mobile Sensing Devices

The model considered throughout the article for the support of end-to-end communications and security with mobility is illustrated in Figure 1. In this model we consider the existence of two or more 6LoWPAN WSN under the same administrative domain, interconnected with the Internet via 6LoWPAN border routers (6LBR). As illustrated, sensing devices are free to move between WSN in the same administrative domain.

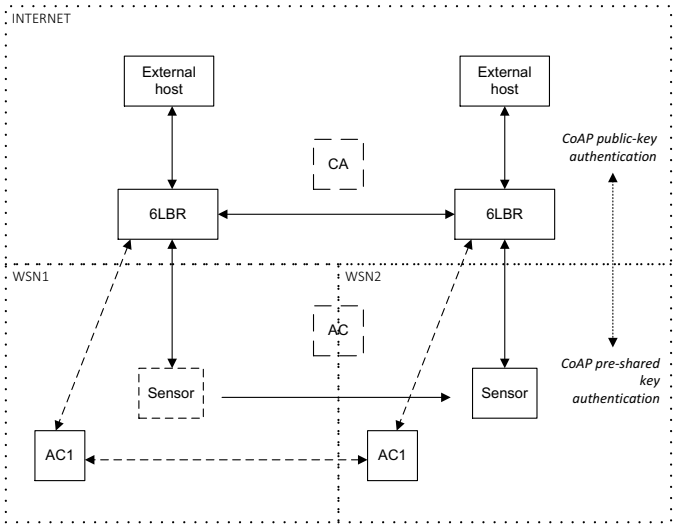


Figure 1 – An integration model for transparent end-to-end transport-layer security with Internet-integrated mobile sensing devices

As illustrated in Figure 1, end-to-end communications and security sessions can be established at the transport-layer, between external (Internet) entities and a mobile sending device, via any of the 6LBR in the scenario. As we discuss in detail in the next Section, a 6LBR is able to transparently intercept and mediate the DTLS authentication and key negotiation phase, at the same time supporting the role of CoAP proxy. As is visible in the previous figure, we consider the usage of different authentication strategies, in the context of a single end-to-end session. From the perspective of an external entity this session is being authenticated using the CoAP security mode providing the highest security: public-key authentication with certificates

(using the CoAP *Certificates* [4] security mode). On the other hand, mobile sensing devices may employ a much lighter and realistic authentication strategy: based on pre-shared keys (using the *PreSharedKey* CoAP security mode). We must note that our goal is to be able to this with total transparency to the communicating parties, and thus, neither the sensing device nor the Internet client are aware that the other party doesn't support the same authentication strategy.

Another important aspect of the proposed integration model can be found in the support of mobility, which works in tandem with a protocol we introduce to support authentication in the WSN domain. For the purpose of supporting authentication and mobility, we consider the usage of a Certification Authority (CA) and of Access Control (AC) entities. The CA attests the validity of the various communicating entities by issuing certificates, while AC servers assist in mobility and authenticating the WSN communicating parties, in the context of end-to-end Internet communications. Although in the previous figure AC servers apply to a particular WSN domain, we can also consider multihoming, with a single AC server supporting authentication and mobility for the various WSN in the administrative domain.

In order to provide effective security, we need to address the trust model considered in the integration scenario. Trust is established between AC servers on different WSN domains, in order to support end-to-end security with mobility, as we discuss later in the article. We also assume that a 6LBR trusts its AC server, the same applying to the mobile sensing device. Trust is configured in the form of shared cryptographic keys during the configuration or network bootstrap phase, as we discuss later in the context of the proposed authentication and mobility procedures. Finally, we also assume that the 6LBR, AC and CA devices are without the constrains in resources of mobile sensing platforms, and thus are able to support the proposed end-to-end security, authentication and mobility mechanisms. Regarding the threat model considered, we note that our focus is on providing security against external attacks, and in particular in enabling fundamental security properties as confidentiality, integrity, authentication and non-repudiation to end-to-end communications with constrained sensing devices, using the mechanisms we proceed to describe.

IV. Mechanisms towards transparent end-to-end security with mobility

As per the goals of this article, in the context of the interconnection model previously discussed we propose a mechanism to assist in the support of effective end-to-end security, in the presence of mobile sensing devices. We begin by describing our approach to DTLS transparent interception and mediation, and later we present the protocol responsible for the support of authentication and confidentiality in the WSN part of the end-to-end security session. Finally, we address the support of mobility between WSN domains. Overall, it is our goal that the proposed mechanisms work in tandem to provide effective end-to-end security with mobility, in a completely transparent fashion to communicating parties and applications, and at the same time with total compatibility with CoAP security as current defined for the IoT.

A. DTLS transparent interception and mediation

The first challenge we address is to release constrained sensing devices from the burden of having to support costly ECC computations in the context of the initial DTLS handshake. We must note that the handshake is the problematic part of end-to-end security, as after authentication and key negotiation end-to-end security may be addressed in the sensing device efficiently, if AES/CCM encryption is employed. As we also note later in the article, the transparent interception and mediation of DTLS also provides advantages other than the enabling of ECC encryption to support high security with CoAP.

The DTLS handshake is an important part of end-to-end security, as it allows for mutual authentication and key agreement between both communicating parties. Not only we want to offload such costly computations, we want to do it in a completely transparent fashion to such parties and applications. We also need to support sensing devices that may freely move between different WSN domains, as previously discussed and illustrated in Figure 1. We guarantee that, in the context of a given IoT application, CoAP resources residing on sensing devices are reachable securely, irrespective of the current position of the device, and at the same time not requiring any modification to CoAP and DTLS as supported on such devices. The preservation of total compatibility with DTLS and CoAP specifications is of cornerstone importance in our proposal.

The proposed mediated DTLS handshake supports delegated ECC public-key mutual authentication between mobile sensing devices and other external (Internet entity), as illustrated in Figure 2. We note that the interception of the DTLS handshake at the 6LBR allows us to control how the handshake is performed with the two end parties, in a completely transparent fashion to such entities. Thus, from the point of view of the Internet client and CoAP server (as considered in Figure 2), the handshake is performed accordingly to the rules defined for DTLS [6], which basically adapts TLS (Transport Layer Security) [10] for performing over UDP (as employed in 6LoWPAN environments).

Considering the integration model illustrated in Figure 1, the interception and mediation of messages is performed in the 6LBR, which also supports Internet communications between the WSN and Internet domains and, if required, a CoAP proxy in either reverse or forward mode. On the Internet side, a CoAP client wants to retrieve information from the CoAP server running on the sensing device and connects via the 6LBR. Such communications are intercepted at the 6LBR and the router is able to expose authentication and key negotiation differently towards the WSN side, in communications with the sensing device. We also allow the opposite usage scenario, meaning that the client may be on the WSN domain connecting to a CoAP server on another WSN network or on the Internet. As illustrated, AC servers are also part of the handshake for the purpose of supporting authentication between the 6LBR and the sensing device.

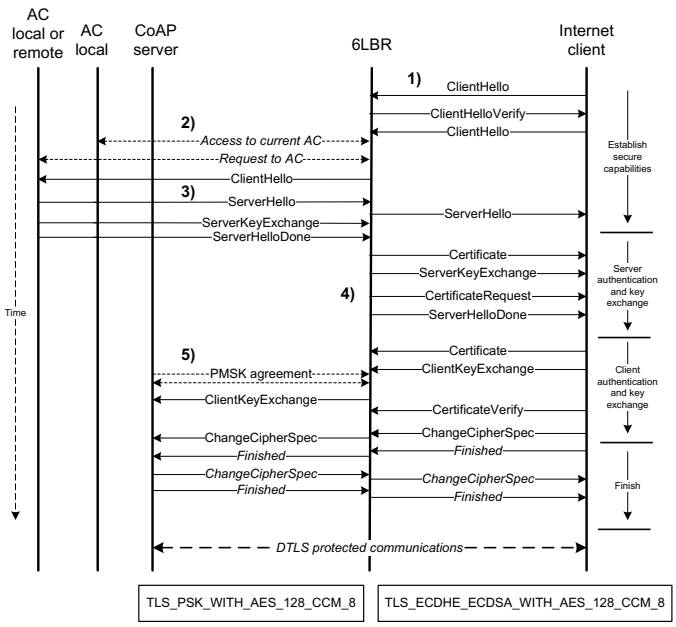


Figure 2 – Transparently mediated DTLS handshake with mutual authentication and mobility support

On the Internet side we allow for the usage of authentication using ECC cryptography and certificates, thus supporting the *Certificates* CoAP security mode [4,5], while on the WSN we employ the much lighter *PreSharedKey* CoAP security mode, certainly more aligned with the real capabilities of constrained sensing devices. In line with the CoAP security modes supported, on the Internet side we consider the usage of TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, while on the WSN domain we employ pre-shared key authentication with TLS_PSK_WITH_AES_128_CCM_8.

A DTLS security session requires the two communicating parties to agree on the cipher suite and encryption keys employed. The DTLS handshake transports the information required to derive such secret keying material. The encryption keys required to secure transport-layer communications are obtained from a master key that the client and server must share after the completion of the handshake and, on the other hand, this master key is obtained by both parties using a pair of client and server random values plus a pre-master secret key. We must note that client and server random values are exchanged during the handshake, while the way the pre-master shared key is obtained depends on the cipher suite employed. With cipher suites employing public-key authentication, the client is allowed to generate the pre-master shared key and send it to the server encrypted with the server's public-key. Thus, this is true for TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, which we support in communications with the Internet side, while for pre-shared key suites (TLS_PSK_WITH_AES_128_CCM_8) this is not supported, mainly because at an initial stage the two entities are unable to support the secure transmission of the pre-shared secret. In order to circumvent this limitation, TLS_PSK_WITH_AES_128_CCM_8 is modified in our proposal in order to allow the 6LBR to transmit the pre-master secret to the sensing device. The pre-master secret key received from the Internet client is forwarded to the CoAP server, and in order to guarantee appropriate security for WSN communications, we introduce an authentication and encryption protocol, described later in the article.

Referring again to Figure 2, the following are the main phases or message flights of the mediated DTLS handshake:

1) The initial *ClientHello* is intercepted by the 6LBR, and the router answers with a *ClientHelloVerify* as a measure of protecting the WSN domain against DoS attacks [6]. The *ClientHello* message returned by the Internet client transports the client random value, together with the protocol version and the list of supported cipher suites.
2) Using the proposed WSN authentication protocol (discussed later in the article) the 6LBR obtains an initial ticket from the AC server together with information about the AC to contact for the purpose of obtaining access to the destination sensing device. Here the AC in the same WSN domain as the 6LBR is identified as the "local AC", whereas the AC in the domain to which the sensing device is currently attached to is the "remote AC". From the remote AC the 6LBR obtains a ticket for the CoAP service, information about the cipher suites supported by the sensor, as well as its digital certificate and current IPv6 address.
3) The original *ClientHello* message is forwarded to the destination CoAP device with a request for pre-shared key-based authentication. The *ServerHello* response is forwarded back to the Internet client, this time acknowledging public-key authentication. The *ServerKeyExchange* message forwarded in this flight transports the server random value.
4) In order to guarantee mutual authentication as per our goals, the 6LBR client is authenticated by requesting its certificate. The *ClientKeyExchange* message sent by the client transports the random value and the pre-master secret key generated by the client.
5) The WSN authentication protocol allows us to obtain a secret key to be shared between the 6LBR and the destination CoAP sensing device. We use this key to secure the transmission of the pre-master secret key to the server. The next message flight allows to finalize the handshake between the client and device. After this stage end-to-end communications proceed normally, and the 6LBR is also in possession of the required cryptographic material to support other security mechanisms, as we address at the end of the article.

The WSN authentication and confidentiality protocol is of major importance in the proposed mediated DTLS handshake. This protocol not only supports mobility by informing the 6LBR of the current position and of the AC responsible for the destination sensing device, but also guarantees appropriate high security for WSN communications between the 6LBR and that

device, in the context of the handshake. After the destination sensing device has received the *ClientKeyExchange* message, both communicating parties are now in possession of the same pair of random values and pre-master secret key. This is the information required for both parties to compute the DTLS master key as in the current specification [6], and from this master key to obtain the secret material for DTLS security.

B. WSN authentication and confidentiality

The authentication and confidentiality protocol proposed is responsible for guaranteeing appropriate security in the WSN domain, during communications between the 6LBR and the destination sensing device in the context of the handshake. This protocol also plays an important part in the support of mobility. We illustrate the proposed protocol in Figure 3, noting that it inherits characteristics from the Kerberos authentication protocol [11], while supporting other characteristics designed to support our end-to-end mediation approach, as well as mobility.

As in Kerberos, this protocol considers the usage of two security-related data structures: tickets and authenticators. In generic terms, considering a client named *c* and a destination service named *s*, a ticket $T_{c,s}$ and an authenticator A_c are defined as follows:

$$T_{c,s} = \{ s, c, addr_c, timestamp, life, K_{c,s} \}K_s$$
$$A_c = \{ c, addr_c, timestamp \}K_{c,s}$$

A ticket authenticates a client to a service, in our authentication protocol to authenticate the 6LBR to the remote AC server and to the final CoAP service running on the sensor. As the ticket is opaque to the client, it is transmitted as is to its destination. An authenticator is generated by the client and allows security against replay attacks. The following are the main phases of the authentication protocol:

1) The 6LBR requests, from its local AC server, a ticket and information about the remote AC. The remote AC is the server to contact to request a new ticket for the destination CoAP service.

2) The 6LBR contacts the remote AC server and requests a ticket for the destination CoAP service. This reply, in addition to the ticket itself, transports information on the capabilities of the sensor, its certificate and the current IPv6 address.

3) Finally, the 6LBR authenticates with the destination CoAP service. After authentication, the 6LBR and the sensor share a secret key that they use to secure the transmission of the pre-master shared key, in the context of the DTLS handshake.

As already referred, we assume that trust is established between the various communicating parties previously to communications and end-to-end security. As illustrated in Figure 3, secret keys are shared and used to secure communications between the 6LBR and the AC server ($K_{c,ac}$) and between the AC server and the constrained sensing device (K_s). Trust relationships are also established between AC servers on different WSN domains. This is required to extend the trust model and security from one WSN domain to another, as required to support mobility. Such keys allow a client to obtain, from its local AC server, a key to request, from a remote AC, a ticket for the destination CoAP device. We also assume that, contrary to communications to and from sensing devices, communications between 6LBR, AC and CA entities run over a communications medium without the limitations of the WSN. For each registered sensor the AC servers store its X.509 ECC certificate, the list of supported ciphers and compression methods, the name of the AC server for the WSN domain where the sensing device is currently located, and its current IPv6 address.

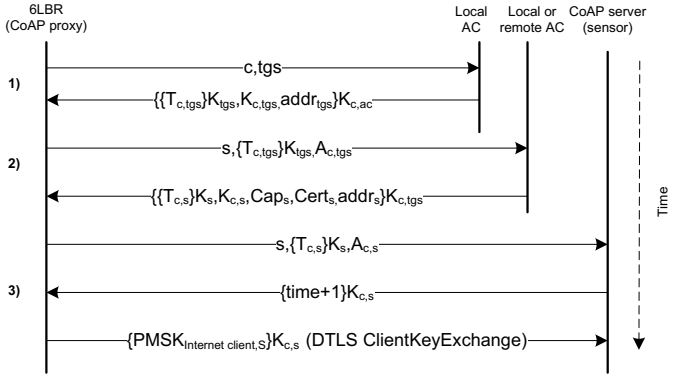


Figure 3 - Authentication protocol for mutual authentication and confidentiality in the WSN domain

The list of supported ciphers allows our model to be applied with other ciphers on the WSN side, although we are currently considering, in this article, the usage of TLS_PSK_WITH_AES_128_CCM_8, as previously discussed. The certificate represents the device and this model alleviates the device from the burden of having to store it in its memory, given that we are performing ECC computation on the 6LBR via delegation. Compression negotiation is supported by the DTLS handshake and also with the mediated DTLS handshake. The name of the AC server of the network to which the sensing device is currently attached to, together with its current IPv6 address, allow any 6LBR to remotely contact the device and activate end-to-end security, as required for devices that are mobile.

C. Support for inter-WSN mobility

One main motivation of our proposal is to address mobility in the context of transparent end-to-end security. In this context, we consider the mobility model illustrated in Figure 4. In this model a sensing device is free to move between different WSN domains (inside the same administrative domain) while being able to accept and maintain active end-to-end security

associations at the transport-layer, transparently from the point of view of applications.

For the purpose of dealing with security, we consider the support of mobility side-by-side with network configuration and ND (Neighbor Discovery) procedures, as currently defined for 6LoWPAN [12]. In this context, a change in the IPv6 address of a sensing device, either due to movement, or when the device wakes up in a different WSN, is fired up by the procedures defined in the context of ND. Such procedures may be related with Neighbor Unreachability Detection (NUD), the reception of a Router Advertisement (RA), or in consequence of a Router Solicitation (RS) message sent. In all situations, the IPv6 address of the sensor is updated, based on its link-local address. The mobility model illustrated in Figure 4 consists of the following main phases:

1) A change in the IPv6 address of the device takes place, based on its link-local address and according to ND procedures optimized for 6LoWPAN [12]. In this context, ND messages are exchanged between the device and the 6LBR, in particular RS, RA, Neighbor Solicitation (NS) and Neighbor Advertisement (NA).

2) The 6LBR is responsible for updating information on the new location of the sensor in the local AC server.

3) The 6LBR is also responsible for informing other 6LBR on the new location of the device. For this purpose, we assume the usage of a broadcast-capable shared communications medium.

4) AC servers in the remaining WSN domains under the same administrative domain see their information on the sensor updated by its local 6LBR.

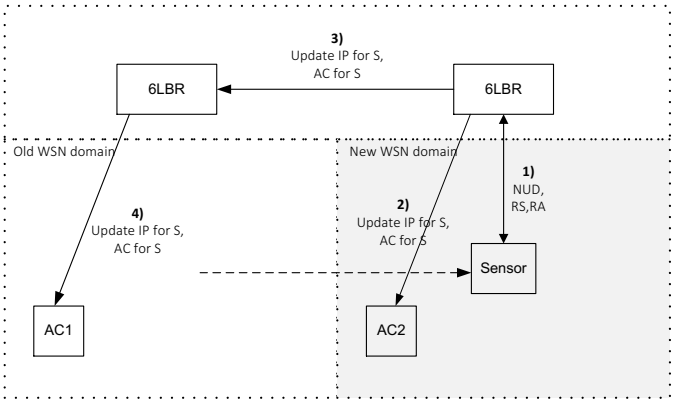


Figure 4 – Mobility and update of information regarding the sensor's current network of attachment

As per the mobility model and the authentication protocol previously discussed, the information on the current position of the sensing device is updated and stored in the various AC servers of the domain. We note again that such procedures also apply in case a multi-homed AC server is employed.

V. Experimental evaluation

We evaluate the previously described security and mobility mechanisms experimentally, looking in particular for two aspects we consider critical for the effectiveness of any proposal on security for constrained wireless sensing environments: the impact on energy and computational effort.

A. Experimental evaluation setup

The integration model illustrated in Figure 1 is considered again, this time for the purpose of evaluating the proposed mechanisms. For this purpose, we employ TelosB [7] sensing devices running the TinyOS operating system [13], and also Linux hosts, for the roles of 6LBR, AC, CA, and Internet client. We employ TinyOS with support for the 6LoWPAN stack, CoAP and also the proposed security and mobility-related procedures. For the purpose of symmetric encryption, we also benefit from the usage of standalone AES/CCM encryption available at the hardware in the TelosB, using code appropriate for this purpose [14]. ECC cryptography is supported using code based on TinyECC [15], and the Internet CoAP client uses *libcoap* [16] integrated with DTLS. Measurements on energy were obtained by measuring the voltage across a current resistor, placed in series with the battery pack of the sensor, while the computational effort was derived directly from the system clock of the sensing device.

B. Lifetime of sensing applications

Our first goal is to evaluate the impact of the proposed mechanisms on energy, as this may directly dictate the potential lifetime of the device and consequently any IoT application depending on it. We measured the energy required to support applications employing the mediated DTLS handshake with sensing devices moving between different WSN domains. For both aspects, we measure energy required for processing headers, security and communications, considering the employment of 102-bytes 6LoWPAN packets. As per our evaluation, the proposed mediated DTLS handshake requires a total of 20 6LoWPAN messages (including the messages required for the WSN authentication protocol) and a total of 0.0013 mJ (millijoules) from the energy available in the TelosB. As expected, the original DTLS handshake is much more demanding, as it requires a total of 39 6LoWPAN messages and 54.4 mJ of energy from sensing devices. Regarding DTLS encryption with standalone AES/CCM on the sensing device, it requires 0.0002 mJ, in deep contrast with 10.89 mJ required to support public-key ECC digital signing, as required with TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8. We may clearly observe the impact of ECC on constrained sensing devices with the characteristics of the TelosB. We note that the previous values are total, measured from the reception of a 6LoWPAN packet to the time when cryptography finished processing the packet on the sensor. As such, we are capturing the total energetic effort to process end-to-end security for a packet. We also consider the energy required for the processing of a packet and related security headers, measured as 0.007 nJ (nanojoules).

Another important aspect considered in our evaluation is the impact of the proposed mobility model and procedures on the energy available on sensing devices. For this purpose, mobility is conjugated with ND mechanisms as previously discussed, and we evaluate the energy required for ND, reception of information from the 6LBR and derivation of a new IPv6 address from the link-local address of the device. Regarding security in the context of ND, we consider the usage of AES/CCM encryption to protect ND-related messages, as defined in [12]. Overall, the total energy cost of supporting communications and security on the TelosB was measured as 0.001mJ.

The previously discussed values obtained experimentally allow us to derive analytically the predictable lifetime of an application using the proposed security and mobility mechanisms. Without considering mobility, it is clear that the proposed mediated DTLS handshake always provides greater lifetime values [8], given that in the original DTLS handshake sensing devices are required to support costly ECC computations during session establishment.

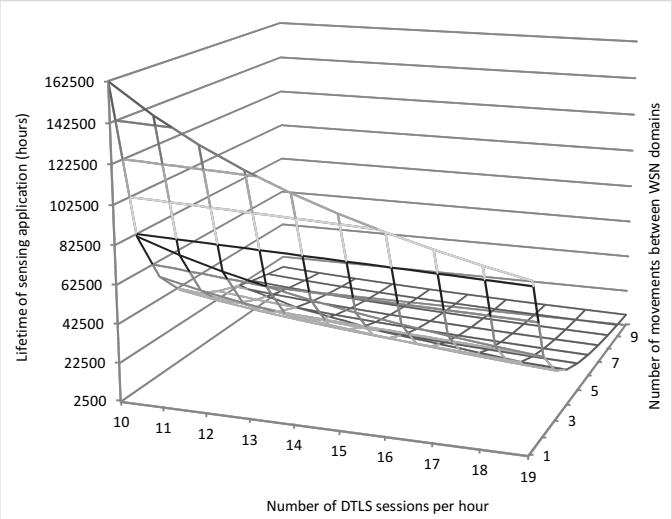


Figure 5 - Impact of end-to-end security on the lifetime of sensing applications, with mobility

We next evaluate the impact of the delegated DTLS handshake in conjugation with mobility, as illustrated in Figure 5. In this figure we illustrate the predictable lifetime (in hours) in respect to the number of DTLS sessions established with a sensing device, and also to the number of movements of the device between WSN domains. We also consider that a CoAP request (consisting of two 102-bytes 6LoWPAN packets, one containing a confirmable request and the other the corresponding reply) is served every time the sensor moves to a new WSN. The values represented in Figure 5 consider the usage of the TelosB sensing device powered by two new AA LR-6 batteries.

Even for the worst scenario (in this case 19 DTLS sessions per hour, 10 movements between WSN domains per session and 1 CoAP request per visited WSN) the expected lifetime remains above 8500 hours. It is clear that for less demanding scenarios in respect of mobility, we are able to obtain much better values. For example, when considering 14 DTLS sessions per hour and 5 movements between WSN domains, the predictable lifetime is around 23 thousand hours, thus 3 times over the previous calculation. We also observe an expressive decline in the expected lifetime when mobility requires more changes in the WSN, during the lifetime of a DTLS session. This is due to the fact that we are securing mobility-related communications with AES/CCM, and as the number of movements increases the impact of AES/CCM security is larger than that of supporting the DTLS handshake. Overall, from our previous evaluation, we are able to confirm that the proposed security and mobility mechanisms are able to provide viable lifetime values is all of the considered usage scenarios.

C. *Maximum communications rate*

Wireless sensing devices as the TelosB don't possess mechanisms such as multi-threading, and as such the computational time required to support security directly influences the maximum communications rate that a sensing device may support. IoT applications may thus suffer if security is too resource demanding, also from the perspective of its computational requirements.

As for energy, we experimentally measure the computational time required to support the proposed mechanisms. As expected, the time required to support the mediated DTLS handshake (15.39ms) is much lower than to support the original handshake (10.09s), again due to the computational impact of ECC [8]. We are able to analytically derive the maximum number of CoAP requests per hour that a device is able to sustain, in the presence of end-to-end security and mobility, as illustrated in Figure 6.

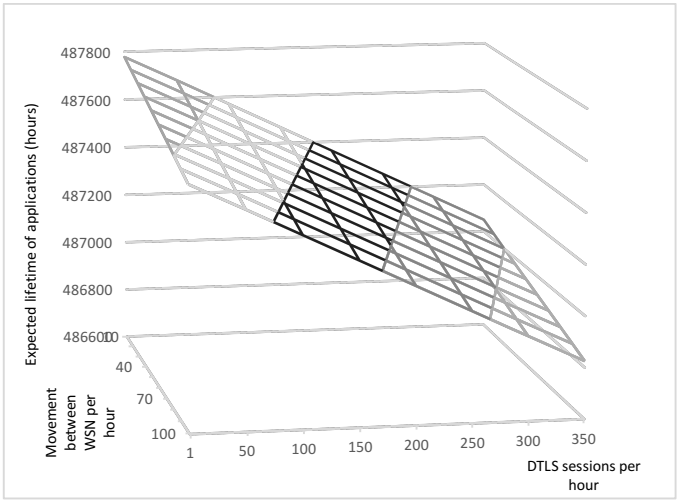


Figure 6 - Impact of end-to-end security and mobility on the communications rate of sensing applications

We must note that mobility also impacts on CoAP communications, as while processing mobility-related procedures the sensing device is unable to accept and serve CoAP requests. As we may observe in Figure 6, the proposed security and mobility mechanisms are able to still guarantee appropriate communication rates. We may note that for

example in the worst case scenario (as considered, for 350 DTLS sessions established per hour, or around 10 sessions per second, and 100 movements of a device to a new WSN domain), a sensing device would be able to still serve over 480 thousand requests during one hour, or over 133 per second. We may objectively consider this to be well above what would be required from a sensing device constrained in terms of energy, in a real application scenario. If one compares end-to-end security with mobility against the usage of DTLS as currently proposed we observe that, even with the added cost of dealing with mobility, the solution proposed in this article performs better. Due to the cost of supporting ECC encryption, the original DTLS handshake as proposed for CoAP is only viable up to 356 DTLS sessions per hour.

VI. Related Work

The employment of DTLS to secure CoAP communications raises various issues, as addressed throughout the article, that are also recognized and the focus of research. As previously discussed, ECC as proposed to provide security to CoAP is too resource demanding, and in this context alternatives approaches are being proposed. The impact of ECC cryptography, as well as the efficiency of AES/CCM to support pre-shared key authentication, is also verified in other works [17][18][19]. Other aspects complicating the adoption of DTLS are the need to store and process public-keys and certificates on constrained sensing devices, and the inadequateness of the protocol when CoAP proxies are employed. As previously observed, those are aspects that also contribute to motivate our approach.

Some authors address the proposal of usage profiles for DTLS, in order to better cope with the employment of 6LoWPAN and the characteristics of constrained wireless sensing platforms, as in [20]. Others propose modifications to the standard itself, for example the adoption of 6LoWPAN IPHC compression as a way to reduce the size of DTLS headers [21]. An alternative proposal in this context consists in the usage of CoAP communications to support costly DTLS handshake operations [22]. Overall, such proposals to do not solve the problem of effectively supporting ECC-based authentication and key negotiation on constrained sensing devices, nor address the need to cope with mobility.

More close to our approach in this article, authors in [23] propose a mechanism based on a proxy to support sleeping devices. In this work a mirroring mechanism is employed to serve data on behalf of sleeping smart objects. We also note that this proposal does not offer a solution to address true end-to-end security, the same applying to mobility. In [24] an end-to-end architecture supporting mutual authentication with DTLS is proposed, employing specialized trusted-platform modules (TPM) supporting RSA cryptography on sensing devices. Thus, RSA is adopted with the help of specialized hardware devices, rather than supporting ECC public-key cryptography as currently required for CoAP. Although this proposal addresses end-to-end security, it does not provide compatibility with the current CoAP specification nor does it address mobility.

Overall, we observe that none of the previous proposals offers a solution to effectively support ECC cryptography in the context of end-to-end DTLS security with Internet-integrated sensing devices, in a transparent fashion to the communicating entities and applications, and also supporting mobile devices.

VII. Conclusions and Future Work

In this article we propose mechanisms for the support of end-to-end security with Internet-integrated mobile sensing devices, in the context of an integration model that, in practice, supports various usage scenarios and applications. As previously discussed, we focus on addressing three important aspects that, in the context of real applications, difficult the employment of CoAP with end-to-end DTLS security. One is to offer an effective and transparent solution to the problem of supporting ECC authentication and key agreement, one important goal to support CoAP communications with a high degree of security. Other aspect is the incompatibility of DTLS with the usage of CoAP proxies, which may be supported at the security gateway (6LBR) in our model, while also implementing other security policies. Finally, we also address mobility, and propose a way to abstract end-to-end communications and security from the movement of sensing devices.

The proposed mechanisms were evaluated experimentally considering two main aspects: the impact of such mechanisms on the energy of sensing devices, and also the computational cost. We consider such two aspects to be fundamental in evaluating the effectiveness of any proposal on security for constrained wireless sensing platforms.

It is our goal that the proposed security and mobility mechanisms may provide useful contributions, in the context of the communications and security stack currently being formed to support future IoT applications. As future research objectives, we will target the design of additional security mechanisms based on the integration model considered in this article. One aspect we plan to focus on in the near future is that of intrusion detection or content filtering for CoAP communications. From the proposed mediated DTLS handshake we note that, after the handshake has finalized, the 6LBR may also be in possession of the security data required to compute the cryptographic material used for end-to-end security with DTLS. This opens to door to the design of filtering or intrusion detection mechanisms for CoAP, based on 6LBR devices that, by nature, are placed strategically to protect Internet-integrated WSN domains from abusive CoAP requests or other external threats.

References

1. Kushalnagar N et al. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. *RFC 4919*, 2007.
2. Montenegro G et al. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. *RFC 4944*, 2007.

3. Hui J et al. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. *RFC 6282*, 2011.
4. Shelby Z et al. Constrained Application Protocol (CoAP). draft-ietf-core-coap-13, 2012.
5. Shelby Z. Constrained RESTful Environment (CoRE) Link Format. *RFC 6690*, 2012.
6. Rescorla E et al. Datagram Transport Layer Security Version 1.2. *RFC 6347*, 2012.
7. TelosB Mote Platform, http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf (accessed Mar 2016).
8. Granjal J, Monteiro E and Silva J. "End-to-end transport-layer security for Internet-integrated sensing applications with mutual and delegated ECC public-key authentication." *IFIP Networking Conference, 2013*. IEEE, 2013.
9. Granjal J, Monteiro E and Silva J. "Security in the integration of low-power Wireless Sensor Networks with the Internet: A survey." *Ad Hoc Networks* 24 (2015): 264-287.
10. Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol, Version 1.2. *RFC 5246*, 2008.
11. Neuman B, Ts'o T. Kerberos: an authentication service for computer networks. *IEEE Communications Magazine,* 1994, 32(9), 33-38, DOI: 10.1109/35.312841.
12. Shelby Z et al. Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). *RFC 6775*, 2012.
13. TinyOS Operating System, http://www.tinyos.net/ (accessed Mar 2016).
14. Standalone hardware AES Encryption using CC2420, http://cis.sjtu.edu.cn/index.php/The_Standalone_AES_Encryption_of_CC2420_(TinyOS_2.10_and_MICAz) (accessed Mar 2016).
15. Liu A, Ning P. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. Proceedings of the 7th h international conference on Information processing in sensor networks (IPSN '08), 2008.
16. LibCoAP, http://sourceforge.net/projects/libcoap/ (accessed Mar 2016).
17. De Meulenaer, G. et al. On the energy cost of communication and cryptography in wireless sensor networks. Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,. IEEE, 2008.
18. M. Botta, M. Simek and N. Mitton, "Comparison of hardware and software based encryption for secure communication in wireless sensor networks," *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*, Rome, 2013, pp. 6-10. doi: 10.1109/TSP.2013.6613880
19. Raza, Shahid, et al. "Secure communication for the Internet of Things—a comparison of link-layer security and IPsec for 6LoWPAN." *Security and Communication Networks* 7.12 (2014): 2654-2668.
20. Tschofenig H, Fossati T. TLS/DTLS Profiles for the Internet of Things. Constrained Application Protocol (CoAP). draft-ietf-dice-profile-17.txt, 2015.
21. Shahid R, Daniele T and Voigt T. 6LoWPAN compressed DTLS for COAP, 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), 287-289 2012 doi: 10.1109/DCOSS.2012.55.
22. Brachmann M et al. End-to-end transport security in the IP-Based Internet of Things. 21st International Conference on Computer Communications and Networks, 1-5 2012 doi: 10.1109/ICCCN.2012.6289292.
23. Sethi M, Jari A and Ari K. End-to-end security for sleepy smart object networks, 37th IEEE Local Computer Networks Workshops, 964-962 2012 doi: 10.1109/LCNW.2012.6424089.
24. Kothmayr T et al. DTLS based Security and Two-Way Authentication for the Internet of Things, Ad Hoc Networks, 11 (8) 2710-2723 (2013) doi: 10.1016/j.adhoc.2013.05.003.

User Location Tracking Attacks for LTE Networks Using the Interworking Functionality

Silke Holtmanns
Bell Labs - Nokia Networks
Espoo, Finland
Email: silke.holtmanns@nokia.com

Siddharth Prakash Rao
Department of Computer Science, Aalto University
Espoo, Finland
Email: siddharth.rao@aalto.fi

Ian Oliver
Bell Labs - Nokia Networks
Espoo, Finland
Email: ian.oliver@nokia.com

***Abstract*—User location tracking attacks using cellular networks have been known since 2008. In 2014, several Signalling System No 7 (SS7) protocol based location tracking attacks were demonstrated, which particularly targeted the cellular roaming in GSM networks. Currently, the mobile network operators are in a gradual process of upgrading to Long Term Evolution (LTE) networks, in addition to replacing SS7 by its successor - Diameter protocol. Though Diameter seems to be an improvement over SS7 in terms of security with the use of IPsec/TLS and certificate based authentication, they still need to communicate with their roaming partners who use less secure SS7. In this paper, we will briefly present the translation of existing SS7 attacks into Diameter-based attacks in LTE networks (under certain assumptions) using Interworking Functions(IWF) - which allows communication between networks that use different protocols. The key contribution of this paper is the the detailed explanation of novel attack vectors to obtain the user location information using IWF and hence, the proof that even new LTE network can be vulnerable to legacy attacks. Furthermore, we will outline some of the potential protection approaches for the attacks that we discuss.**

***Keywords*—Signalling System No.7 (SS7), Diameter, Interworking Funcation (IWF), Location Tracking, Privacy**

I. Introduction

Cellular network technologies require some degree of tracking of user location – specifically user equipment tracking, as part of their fundamental mechanism of working. Without this basic function, features such as hand-over between cells would not work and it is not possible to provide seamless user experience (i.e. no dropped calls or connections) when the user is moving. Furthermore, the aforementioned user tracking by Mobile Network Operators (MNOs) helps to provide cellular services to subscribers of partner MNOs, which indeed is the generic scenario of ”roaming”. In such scenarios, the inter-operator network connection which is used for exchanging information is often termed as the interconnection. Recently, these interconnections have been exploited to track individual subscribers by hackers, especially when the interconnections are bound by Signalling System No.7 (SS7) protocol. In this paper, we describe attack scenarios again targeting such interconnection networks, however, instead of exploiting the GSM networks like the previously found attacks, we exploit the newer generation of mobile telecommunications technology i.e Long Term Evolution(LTE) or 4G. The fundamental idea of this paper is that – an attacker poses as a roaming partner having an old network (SS7) and therefore, forces the new LTE network to use less secure legacy communication messages. We will first walk through the existing attacks and the related work that use SS7, followed by extending those attacks against LTE networks using the Interworking Functions (IWF). We describe this so-called downgrading attack for illegitimate location tracking. This is the first attack published which downgrades the LTE Diameter security to the level of SS7 security over the interconnection.

ISBN 978-3-901882-83-8 © 2016 IFIP

II. Roaming Interconnection

Signalling System No.7 (SS7) is a mobile backend protocol used for interconnectivity between mobile operator networks, which enables roaming and cellular services across operator domains. The protocol is mainly used for communication between the network elements and the networks themselves. It has served its purpose successfully over four decades being a substantial source of income for the service providers and MNOs. In spite of its age, SS7 and its IP version called SIGTRAN continue to be the most commonly used protocols for roaming interconnections till date. In order to provide seamless services to the roaming partners who might have interconnections only over SS7, irrespective of generation of mobile technology (such as GSM, UMTS and LTE), operators are expected to support SS7 protocol. In that sense, all the operators in the world who offer roaming of any type are connected to the older SS7 interconnection network (refer figure 1). Older in this context means that the nodes deployed use the 3rd Generation Partnership Project (3GPP) standards which are older than Release 8.

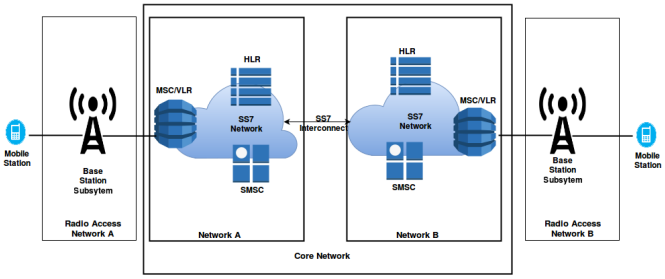


Fig. 1. Two pre-release 8 networks connected via SS7

The Message Application Protocol (MAP) is one of the key applications of SS7 protocol stack which is mainly responsible for the communication between the elements of core network, mobility management and supplementary services. The following elements or nodes of the core network interacts with each other using MAP protocol: (1) Home Location Register (HLR), which contains the subscriber keys and user profile information, (2) Mobile Switching Center (MSC), which manages the user mobility and (3) Visitor Location Register (VLR), which takes care of a user in roaming. Due to the evolution of network technologies and the continuous addition of new services, the MAP specification has grown substantially to support a vast range of services [1].

Unlike the older generation of roaming networks in which subscriber's Home Public Mobile Network (HPMN) (i.e. home network) and Visited Public Mobile Network (VPMN) (i.e. visited network) are connected with SS7 interconnection, the newer LTE networks replaces the SS7 with IP interconnection via the IPX/GRX roaming exchange network. As shown in figure 2, the traffic coming from IPX/GRX interconnection is routed through Diameter Edge Agents (DEA).

As an evolution of HLR, the Home Subscriber Server (HSS) contains the subscriber profiles and it is definitely one of the most important nodes in an LTE network. The Mobility Management Entity (MME) can be seen as the evolution of the MSC, which takes care of the user's mobility management. The home Policy Charging and Rule Function (hPCRF) is the entity that enables billing and thereby collects the charging records for a user. When the subscriber is in a visited network, the same functionality is handled by the visited Policy Charging and Rule Function (vPCRF). The Serving GPRS Support Node (SGSN) handles packet switched data within the network and enables data roaming.

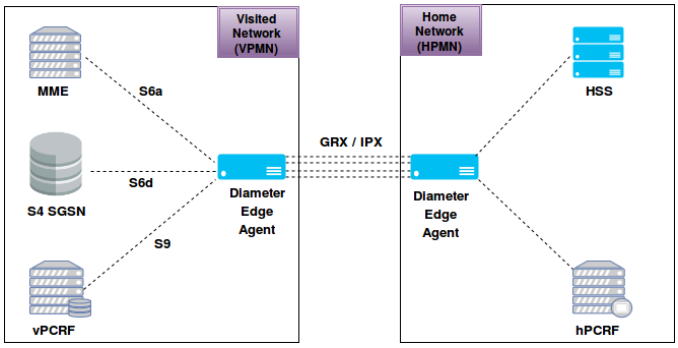


Fig. 2. Diameter roaming implementation between two LTE networks

As mentioned earlier, the upgrade of network (and the supporting infrastructure underneath) from SS7 to Diameter is a gradual process. Most operators update their network infrastructure gradually to avoid service interruption and optimize the return of investment of their infrastructure. During such updates the old equipment are often sold to operators in developing countries, where the capital expenditure is limited and the turnover per user is low. The figure 2 shows the simple direct connection between two operators, both running Diameter. However, the real-life situation is much more complex as shown in figure 3. The number of partners in these cases may scale to around thousand, whose nodes are from different software and hardware releases. The reason for such complex inhomogeneous set-up found in the global interconnection network is either due to the aforementioned gradual update process of supporting network infrastructure or due to limited capital of the operators from developing economies. Irrespective of the reason, such this inhomogeneous set-up provides some interesting attack vectors from security perspective. We describe the exploits using those attack vectors in the subsequent sections.

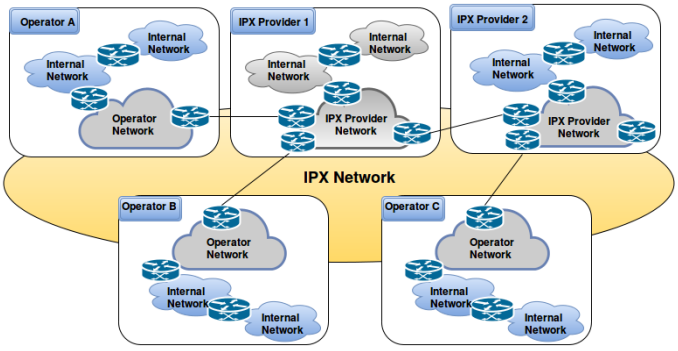


Fig. 3. Roaming hubs and interconnection network

The inhomogeneous set-up simply implies the possibility of existence of nodes within a network that are from different releases and therefore support different protocols. It also implies that the networks towards each other on the roaming interface may use either SS7 or Diameter or the combination of both, depending on the node and the network as outlined in figure 4.

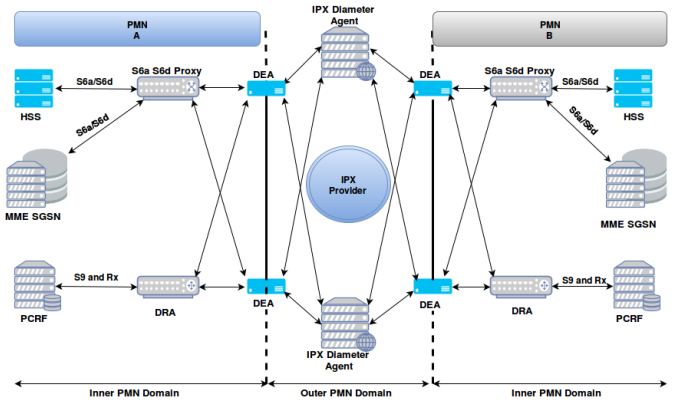


Fig. 4. SS7-Diameter interworking with roaming hubs

For interoperability reasons with their partners, the edge nodes and the nodes themselves have often the ability to translate between Diameter and MAP protocols. Diameter is specified to be secured with NDS/IP [2] (Network Domain Security) security and most commonly IPSec is used as a security protocol. Nevertheless, even the Diameter nodes have

to support partners who run legacy SS7 nodes, where cryptographic security in terms of authentication, confidentiality and integrity is absent.

III. Interworking Function

Usually 3GPP standardizes the functionalities and specifications for communication between nodes belonging to same release. But there are cases, where specific functionality has been standardized to enable interoperability between different releases and technologies. In this realm, the Technical Specification (TS) 29.305 [3] and the non-binding Technical Report (TR) 29.805 [4] describe how Attribute Value Pairs (AVPs) of Diameter and SS7-MAP messages can be mapped to each other. AVPs can be considered as *variables* which often change during cellular communication such as user identity, source of messages etc. Even though this is specified as a feature of mainly edge nodes (e.g. DEA) called Interworking Function (IWF), the way-of-translation is practically deployed on other types of nodes directly to enable interoperability within the operator network, where nodes from different releases are deployed. In such case, where the interworking functionality is used directly at the node, it is often called a multi-domain support scenario. Due to the gradual upgrading within an operator domain, this is a quite common setup as illustrated in figure 5.

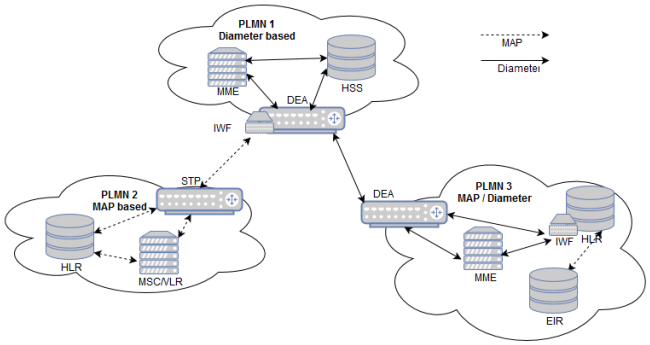


Fig. 5. Three networks with different protocol support

IV. Related Work

Recent successful attacks on SS7 as per [5][6][7][8] and [9] have proven that an attacker with access to the SS7 interconnection network can take control over personal information of the users such as location tracking, billing data and Short Message Service (SMS) messages, in addition to eavesdropping. A detailed explanation about different types of SS7 based attacks can be found in [10] and [11]. We will briefly present only the location-based attacks in this section, followed by investigating in the subsequent sections whether they can be performed on a Diameter network using IWF. On the conceptual level, the idea is to validate whether the Diameter networks are vulnerable to SS7 location attacks by using the IWF for *attack translation*. For all attacks presented in this section we assume that the attacker has access to an SS7 network.

The following location tracking attacks using SS7 are known at the time of writing:

Attack 1: Location disclosure using call setup messages
An attacker here uses the general message flow of a call set up to determine the approximate location of the victim. The attacker with SS7 access pretends to be GMSC (Global MSC), potentially of a partner of the victim's operator, however, the attack may also work with a random GT. The Global Title (GT) uniquely identifies a node in the SS7 network, similar to MAC address in an IP network. The target is a Home Location Register (HLR), the node holding crucial subscriber data.

1) The attacker posing as a GMSC and executes the routine call set up procedure from the point where the GMSC is supposed to receive the Initial Address Message (IAM). At first, he encloses the victim's MSISDN (phone number) in a MAP *Send Routing Information (SRI)* message to the HLR in victim's home network, provided he learns the GT of HLR (It is often found using brute force requests to the GT range an operator holds).
2) The HLR maps the MSISDN sent by the attacker to the International Mobile Subscriber Identity (IMSI), followed by querying the Visitor Location Register (VLR) of victim's visited network by sending MAP *Provide Roaming Number (PRN) Request* to facilitate the call setup. The IMSI is quite important as it is the network internal subscriber identity required by most of the MAP and Diameter commands.
3) The legitimate VLR answers via MAP *Provide Roaming Number* ACK message which in turn contains the IMSI of the victim and the global title of the serving VLR to the HLR.
4) This information is returned via MAP *Routing Information* ACK to the attacker who is impersonating GMSC. The GT of VLR learned here can be used to spot the approximate location of the victim in context.

This attack gives only a rough estimate of the location of a victim, but serves to identify whether he is travelling. Depending on the intention, the travel trajectories could be sufficient for the attacker.

Attack 2: Location disclosure using SMS protocol messages
Similar to the previous attack, the attacker impersonates a Short Message Service Center (SMSC). He pretends to have an SMS waiting for the victim and hence, he requests the MSC/VLR location information in order to deliver it.

1) Pretending to be an SMSC, the attacker sends the MAP *Send Routing Information for SM* (SRI SM) message to the HLR by enclosing the MSISDN of victim.
2) The HLR assumes that the SMSC needs to send an SMS to the provided MSISDN, and thus it replies with the MAP *Send Routing Information for SM* ACK message, which contains the IMSI of the victim along with GT of the MSC/VLR that is currently serving the victim.

Usually, an operator has several MSCs deployed in his network, where each MSC is responsible for a large region. Therefore, this attack allows to identify the region where the victim is currently located, similar to the previously described attack scenario.

Attack 3: Location disclosure using CAMEL location management function messages

The attacker in this scenario exploits the fact that, many network nodes do not check whether a message over the interconnection is internal to the network or otherwise. The *MAP Any Time Interrogation* (ATI) message is usually used within the operator network (i.e. internally), thus, it is not intended to be received over the interconnection network. Nevertheless, Positive Technologies [12] and Tobias Engel [9] showed that the ATI command is often successfully answered by an operator even when it is sent via the interconnection. In spite of the fact that many operators started to block ATI commands coming via interconnection after the public revelation of the aforementioned attacks, it is most likely that not every operator in the world would do the same. In the previously described attacks, the attacker at most can learn the GT of victim's MSC/VLR and hence, the attacker can track down the approximate location of the victim. However, using CAMEL protocol [13] messages, the attacker can narrow down the victim's location accurately to a cell ID, which in densely populated areas can be as accurate as to a street address.

1) The attacker impersonates the GSM Service Control Function (gsmSCF) node and sends a *MAP Any Time Interrogation Request* (ATI) message by encapsulating MSISDN of the victim to the HLR.
2) The HLR considers this as a legitimate message and carries it further by sending *MAP Provide Subscriber Information Request* (PSI) message to MSC/VLR of the victim.
3) The MSC/VLR will initiate the *Paging Request* to receive the Cell ID of the victim.
4) This information is handed over to the HLR via *MAP Provide Subscriber Information Response*, and then back to the attacker via ATI *response* message.

Due to increased risk, many operators started to filter the ATI command as mentioned in the attack in context. However, an attacker can bypass such filters by performing a hybrid attack by executing the SMS protocol based attack to know the MSC GT of the victim, followed by sending MAP *Provide Subscriber Information Request (PSI) message* directly to that MSC as described in the ATI based attack. Since the PSI command has a legitimate usage over the interconnection, it is difficult to filter it.

Attack 4: Location disclosure emergency location service messages

Mobile operators are lawfully bound to provide accurate location information of their subscribers during emergency situations such as accidents (initiated by the subscribers themselves e.g. emergence number 911) or criminal tracking (initiated by the operators on behalf of law-enforcement officials. In case of the latter, the operator initiates an internal network command called *MAP Provide Subscriber Location (PSL)*. This command can be exploited for illegitimate location tracking as per the following attack:

1) The attacker needs to know the victim's IMSI and MSC/VLR GT. He can discover those identifiers through SMS protocol or call setup message based attack as described attack 1 or 2 respectively.
2) Now the attacker queries the MSC/VLR in the visited network for the accurate location information of the victim by sending *MAP Provide Subscriber Location Request (PSL)*. In order to do so, the attacker should bypass the Location Service client (LCS) client (in regular circumstances, law-enforcement authorities are the legitimate LCS clients) authentication at the Gateway Mobile Location Center (GMLC), by directly sending the aforementioned PSL message to MSC/VLR. In turn it leaves the MSC/VLR in context with no means of verifying the actual occurrence of the authentication.
3) The MSC/VLR detects the location of victim's mobile station using one of the various possible methods (e.g. *RRLP Request* [14]).
4) The MSC/VLR then responds to the attacker with the MAP *Provide Subscriber Location Response* message, which contains the Cell ID of the location of the subscriber.

The Cell ID can be mapped to a real location in terms of geographic coordinates of the victim, using publicly available web services such as [15]. In some cases, the LCS message might also reveal the closest GPS coordinates of the victim along with the serving cell ID. However, it is not guaranteed to be as accurate as the GPS information provided by the mobile stations themselves (e.g. using any GPS app on the mobile). It should be noted that the aforementioned attack works only when an operator supports the emergency localization feature.

V. Assumptions for the target LTE networks

We assume that an attacker has access to the roaming interconnection network. For an attacker, there are several ways to gain access to the interconnection network:

- Most operators have a wholesale department or subsidiary which rents out access to third parties and various service providers.
- The roaming network is global and it covers countries or regions where having legal access to subscriber data is allowed due to less strict enforcement of the privacy regulations.
- Compromised or misconfigured nodes that are visible on the Internet (e.g. via Internet-connected database such as Shodan.io [16]), could act as the potential points of entry for the evil hackers. Caskun showed the practical feasibility of attacking nodes of a GRX at the DefCon 2015 [17].
- Insider attack (e.g. via social engineering or bribing) can lead to unauthorized access by criminals.

Further details about *how an attacker gains illegitimate SS7 interconnection* is beyond the scope of this paper. However, we assume that an attacker, depending on skills, resources and motivations, finds a way in. On the technical side, we make some general assumptions especially on the configuration of the operator who is under attack as follows:

- IPSec (as per [2]) is not used between the SS7 and Diameter supporting nodes. In other words, the messages are sent in clear text without any cryptographic protection.
- No IP address filtering is done. It should be noted that, even with white and blacklisting filtering methods, some attacks where the attacker uses a compromised partner node, obtains a valid partner IP or uses messages that do not require an answer, may still work.
- No layer matching (comparison and cross/checking of sender and return addresses of different protocol layers) is done on HSS or DEA. In some cases, a direct connection between the roaming partners is absent; instead, the interconnection is mediated by one or several IPX/GRX providers (see figure 3). It should be noted that, in such cases the layer matching cannot be performed. Even if the layer matching is implemented, some attacks still would work because spoofing at different layers could be possible, which eventually allows an attacker to bypass the controls put in place.
- No sanity check is made at the receiving node e.g. check for a preceding message that would be there in a normal message flow.
- The attacker knows the MSISDN (phone number) of the victim and address of the edge node (e.g. DEA).

To many readers, especially those with IT background, the aforementioned assumptions may sound too wide and unlikely to be realistic. However, in the cellular industry where SS7 interconnection with absolutely no security has been working quite well for more than 30 years, those assumptions are confirmed to be realistic: Indeed, they can be found in many operator networks, if not all. Initially when the SS7 network was designed, the interconnection network was intended to be used only by the trustworthy government owned operators and hence, there was no obligation to provide any security. However, at present, due to changes in regulations and opening of the telecommunication backend to new entrants, the number of stakeholders who are connected to the interconnection network is increasing day by day. In this real, the question who will finance the additional costs and overhead for certificate management is highly debatable.

There are no statistics or public information about the number of operator networks that would fall under the assumptions that we have made, as it is more likely that most of the operators are hesitant to disclose whether they are vulnerable. Furthermore, lack of internal network monitoring and security audits, and the recent revelation of the attacks [17] [12] where the similar configurations were exploited, strengthens our assumptions.

In our attack scenario, the operator that is under attack has a LTE network within which he may use NDS/IP security [2]. However, on the interconnection edge, he has a Diameter Edge Agent (DEA), which is collocating an Interworking Function (IWF) corresponding to [3]. Another scenario that works in the similar manner is where the operator does not deploy a DEA. Instead, he connects the nodes directly to the interconnection link and implements the interworking functionality at that node. Such ”direct connection” of important core network nodes are not often, but we speculate that, in future, with Network Function Virtualization the international operators attempt to optimize the usage of their nodes and eventually end up setting the aforementioned direct connections due to ignorance.

VI. Interworking Attack Scenario

The attacks that we describe in this section are the typical downgrading attacks where the the attackers intentionally lower the strong security of a particular protocol or system to that of a much less secure legacy system. Even though such types of attacks are common in the radio access networks [18], they rarely seen in the core signalling systems. As mentioned before, the general idea is that an attacker pretends to a legacy SS7 network or node, thereby forcing the more secure LTE Diameter network to use SS7 MAP protocols for further communication.

The first step for an attacker is to obtain the IMSI of the victim, as the IMSI is one of the primary user identifier needed for majority of the communication within the interconnection network. There are several ways to obtain the IMSI, but, we present an attack vector which use the IWF and describe the procedure to obtain IMSI based on the knowledge of MSISDN in Diameter-based network.

The attacker starts his attack by querying the targeted victim's network using the MAP SRI SM command. However, the success of this attack is guaranteed only in absence of *home routing* and if the Diameter interconnection of the targeted operator is established over the S6c interface with additional support for IWF. The IWF of the targeted network translates the MAP SRI SM to Diameter Send Routing Info For SM Request (SRR) as depicted in figure 6.

The attacker impersonates as a partner SMSC or an IWF (for the two IWFs scenario) in the querying network and claims only to support legacy SS7 MAP by sending the MAP SRI SM request over the interconnection network.

1) The attacker sends a MAP SRI SM request containing the MSISDN to the targeted victim's network. On the underlying protocol layer that facilitates routing (routing is usually facilitated by the Signalling Connection Control layer i.e. SCCP layer of SS7 protocol stack), the attacker can use his own Global Title Calling Party Address (CgPA), since no layer matching is done between SCCP and rest of the MAP layers. In addition to the aforementioned parameters such as victim's MSISDN and cgPA, the attacker requires Service Centre Address (SCA) and set the SM-RP-PRI priority flag in order to

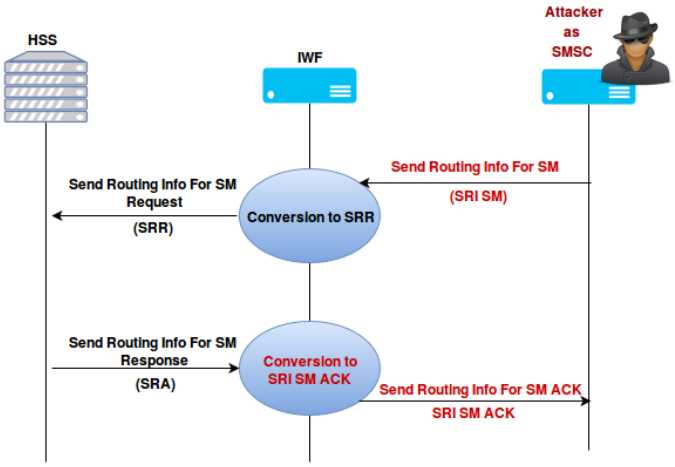


Fig. 6. IMSI disclosure attack using *SRI SM*

craft the MAP SRI SM request command. The attacker can spoof the SCA to hide his identity, whereas the SM-RP-PRI flag enables him to receive relevant information from the HSS of the targeted network even when the victim is not being served by the network in context.

2) The targeted network's IWF receives the MAP SRI SM request and converts it into the Diameter SRR by mapping the received MAP parameters to the corresponding Diameter AVPs. For instance, Diameter AVPs such as SC-Address, MSISDN, and SM-RP-PRI are directly populated based on the corresponding MAP parameters. Whereas the Origin Host/ Realm and Destination Host/Realm AVPs are mapped from the received SCCP CgPA and called party address (CdPA) parameters respectively. More information about these mapping procedures can be found in Annex A.3.5.1 of [3].
3) Once the mapping as described in the previous step is done, the IWF routes the SRR towards HSS of the targeted network via DEA/ DRA. The HSS responds with the Diameter *Send Routing Info For SM Answer (SRA)* command, which contains the IMSI in the User-Name AVP (refer section 5.2.1.1 of [19] for more details) and the nodes currently serving the victim in context. The SRA command is routed back to the IWF again via DEA/DRA.
4) The targeted network's IWF receives the Diameter SRA and converts it into MAP SRI SM response by mapping the received AVPs to the corresponding MAP parameters. The IWF routes the MAP SRI SM response towards the roaming interconnection. If the attack in context is successful, the SRA command contains all the AVPs that an attacker is expecting. In such cases, the IWF populates the MAP SRI SM response by mapping the received AVPs to the corresponding MAP parameters as follow (only the most important parameters are listed):
   - **IMSI** is populated with the value contained in the SRA User-Name AVP.
   - **Network-Node-Number** is populated with the value contained in either of SRA MME Number for MT SMS, MSC-Number, SGSN-Number, or IP-SM-GW-Number AVPs. This field contains the nodes which are currently serving the victim and hence, it can be used by the attacker to launch further attacks or to estimate the rough location either based on MSC or MME number.
   - **Origin and destination Host/Realm** AVPs are mapped to SCCP CgPA of the targeted network's HSS address and SCCP CdPA of the attacker's network address (i.e. the actual GT which enables the attacker to receive the response) respectively.
5) Furthermore, the IWF of the targeted network will send the MAP *Inform Service Center* message to the attacker to confirm the completion of the requested information delivery. However, from the point of view of the attacker, this message is rudimentary, since he would have already received the desired information such as the targeted victim's IMSI, serving node address and possibly the address of the HSS

The aforementioned IMSI retrieval attack is crucial, as the IMSI is used a priori to launch the actual location tracking attacks. This is mainly due to the extensive use of IMSI in Diameter based communication, instead of just the MSISDN or Mobile Station Roaming Number (MSRN) in SS7 based networks. There exist several other ways of obtaining the IMSI, such as using false base station, WLAN access point and EAP-AKA protocol. However, we omit further description about those methods, as they are beyond the scope of this paper.

We now investigate how the four SS7 based location tracking attacks can be extended over a Diameter based network using the Interworking Functions.

Attack 1: Location disclosure using call setup messages The MAP SRI has no direct mapping to Diameter, as is there is no specific entry in the 3GPP standards regarding how the IWF should handle it. This in turn forces the attacker to directly submit the request command in context to the HSS, hoping that the HSS would support a multi-domain scenario. However, the operators rarely connect their HSS directly to the interconnection network, and hence, the success chances of this attack is very unlikely.

Attack 2: Location disclosure using SMS protocol messages using *SRI SM* As mentioned before in the preparation step of an attacker to retrieve IMSI of the victim, the MAP SRI SM message sent to an IWF node retrieves the information about the serving node along with IMSI. Since this attack follows the exact same set of steps of the IMSI disclosure attack, we would skip the repetition of the same. The serving node information in terms of the SRA MME Number for MT SMS in the network configuration of our presumed scenario provides a coarse-grained estimate of the victim's location, specifically at the granularity of MME serving area.

Attack 3: Location disclosure using CAMEL location management function messages

Diameter has no direct mapping of the MAP Any Time Interrogation (ATI) command in IWF related specifications. However, an attacker can perform the hybrid attack of using MAP PSI command as described below, provided he has successfully retrieved the IMSI and serving node information.

1) An attacker poses as an IWF himself (say IWF2) and opens a *MAPv2* channel by sending a *MAP PSI* request to the target network's IWF (say IWF1). This request contains the IMSI and serving node information (i.e. the destination MME serving the victim) which he has previously obtained. In addition to that, the attacker must include the parameters such as *Invocation identity* (which can be a random value) and *Requested Information* (which is set to retrieve *Location information*) [20] [13] in the *MAP PSI* command that he uses for the attack.
   Optionally, the *Requested Information* parameter can include sub-parameters like *Active Location Retrieval requested* and *Location Information in EPS supported*, to obtain more fine-grained location details from the targeted network. The attacker can also request information such as subscriber state, the International Mobile Station Equipment Identity (IMEI), and software version of the victim. For many attackers, the IMEI along with software version is probably one of the most interesting information to have, as they might enable the attackers to launch device specific targeted attacks.
2) The receiving IWF1 of the attacked network converts the *MAP PSI* request into a *Diameter Insert Subscription Data Request (IDR)* as per the mapping guidelines provided in [4] and [3]. During this mapping, the IWF populates *User-Name* AVP based on the IMSI contained in the MAP PSI request and sets the *IDR-flag* to value '3' (this indicates that the location information is requested), along with generating a *Session ID*. The IDR message is then directed to MME/SGSN.
3) The MME/SGSN replies to the IDR command using *Diameter Insert Subscription Data Answer (IDA)* over the SGd/Gdd interface as specified in [21]. Depending on the information requested, the IDA message includes the *EPS User State* and *EPS Location information* AVPs, which contains the subscriber (victim) state and cell ID respectively.
4) On the receipt of IDA message, the IWF 1 translates that into *MAP Provide Subscriber Data Info Ack (PSI ack)* message as per the guidelines specified in the section 8.11.2 of [1]. In particular, the translated PSI ack contains the location information (Cell ID or GPRS information) and subscriber state (if it was requested in step 1).

A variant of the attack in context is when the attacker poses as home-HLR of the victim and sends the MAP *Insert Subscriber Data* command instead of PSI. Even in this case, the MAP specific messages will be translated [11] into IDA/IDR to finally return the MAP *Insert Subscriber Data Ack* (ISD) containing the victim's subscription data.

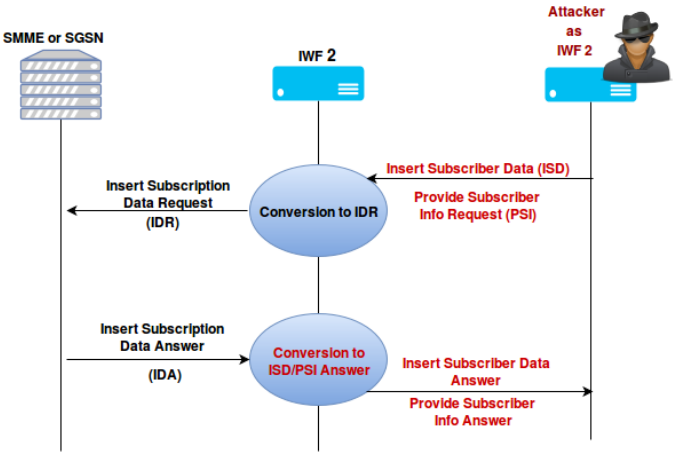


Fig. 7. Location disclosure attack using *MAP PSI*

Attack 4: Location disclosure emergency location service messages

The 3GPP standards for IWF (i.e. [4] and [3]) does not specify the procedures for handling MAP PSL command and hence, the direct mapping to Diameter specific command is not possible. Even if there were relevant specifications, the maximum achievable accuracy of the location is similar to that of the location information retrieved using *PSI command*.

VII. Countermeasures

Interworking with legacy equipment cannot be discontinued without serious service interruption, however, there are several measures which can be deployed in order to improve the security of the interconnection as we describe below:

The first line of defence is the protection of the IMSI, specifically by deploying the home routing for SMS based communication messages, as it makes the IMSI retrieval via the interconnection network much harder. The second layer of defence is to improve the Interworking Function with security features. The Interworking Function should have some additional layers of security, in particular we suggest:

1) Basic SS7 filter or firewall that that effectively verifies whether a message is:
   - network internal or to be received via the interconnection.
   - communicated within the GT range of a contract partner.
   - for an outbound roamer who is actually roaming.
2) Whitelisting of partners and the protocols that they use i.e. an LTE-only partner should use Diameter and not suddenly send a MAP message.
3) Implement NDS/IP security over the Diameter Edge Agents with roaming hubs and with partners who has direct connection along with support for Diameter.

4) AVP specific filtering and modifications e.g. dummy location in MAP PSI over the interconnection.

Diameter security is much closer to the traditional Internet security, which deploys IP based firewalls. In addition, The operators should validate whether the origin realm AVP belongs to one of their partners, and if not, such messages should be either discarded or filtered for further analysis. For the routing level security, the routing need to be based on the origin identity and not on the hop-by-hop identity between nodes to avoid the attack outlined in [22].

VIII. Conclusion

Telecommunications is an intricate system made up of diversified, circuitous subsystems with a multiplicity of different technologies. The complexity increases even further in the worldwide interconnection network which connects operators for the purpose of roaming, as such interconnection contains all possible kinds of legacy systems. Therefore, even the fully fledged LTE operators deploy Interworking Functions to be able to communicate with their partners with legacy technologies. This paper takes the existing SS7 based location tracking attacks into consideration and further investigates the behaviour of the attacks, when they are run against the interconnection nodes with Interworking Function support. Even though some of the attacks fail to harm the LTE networks, the successful attacks that we described, prove the feasibility of translation of legacy attacks on the newer protocols or networks which are believed to be secure. Furthermore, the Interworking functionality may be potentially used to launch other type of attacks such as Denial of Service against a subscriber by using Cancel Location or Purge commands, which is part of our ongoing research.

In conclusion, we argue that the newer generation of mobile networks are vulnerable to legacy attacks. The attacks described in this paper not only outlines a novel evolution of legacy attacks, but also it appeals to be relevant to the current state of telecommunication industry, where the operators are gradually upgrading to networks along with LTE roaming. While, those operators continue to be vulnerable by still supporting some SS7 functionalities until all their roaming partners fully upgrade their networks, the proposed countermeasures are expected to make them relatively secure.

Acknowledgment

The authors would like to thank the GSMA RIFS group members for their drive to improve the security of the global interconnection network and in particular, Looi Kwok Onn for spotting some technical hurdles and suggesting measures to overcome them.

References

[1] 3GPP, "Mobile Application Part (MAP) specification," 3rd Generation Partnership Project (3GPP), TS 29.002, Sep. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/29002.htm

[2] 3GPP, "3G security; Network Domain Security (NDS); IP network layer security," 3rd Generation Partnership Project (3GPP), TS 33.210. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/33210.htm

[3] 3GPP, "InterWorking Function (IWF) between MAP based and Diameter based interfaces," 3rd Generation Partnership Project (3GPP), TS 29.305, Sep. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/29305.htm

[4] 3GPP, "InterWorking Function (IWF) between MAP based and Diameter based interfaces," 3rd Generation Partnership Project (3GPP), TR 29.805, Jul. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/29805.htm

[5] T. Engel, "Locating mobile phones using signalling system 7," in *25th Chaos communication congress*, 2008.

[6] K. Nohl, "Mobile self-defense," in *Vortrag auf dem Chaos Communication Congress 31C3, Hamburg*, 2014.

[7] A. De Oliveira and P.-O. Vauboin, "Worldwide attacks on ss7 network," *FTP: http://2014. hackitoergosum. org/slides/day3_Worldwide_attacks_on_SS7_network_P1security_Hackito_2014. pdf*, 2014.

[8] S. Puzankov and D. Kurbatov, "How to intercept a conversation held on the other side of the planet," 2014. [Online]. Available: http://2014.phdays.com/program/tech/36930/

[9] T. Engel, "Ss7: Locate. track. manipulate," in *FTP: http://events. ccc. de/congress/2014/Fahrplan/system/attachments/2553/or iginal/31c3-ss7-locate-track-manipulate. pdf*, 2014.

[10] S. P. Rao, "Analysis and mitigation of recent attacks on mobile communication backend," 2015.

[11] S. P. Rao, S. Holtmanns, I. Oliver, and T. Aura, "Unblocking stolen mobile devices using ss7-map vulnerabilities: Exploiting the relationship between imei and imsi for eir access," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1. IEEE, 2015, pp. 1171–1176.

[12] "Signaling system 7 (ss7) security report." [Online]. Available: http://tinyurl.com/SS7-Security-report

[13] 3GPP, "Customized Applications for Mobile network Enhanced Logic (CAMEL) Phase X; Stage 2," 3rd Generation Partnership Project (3GPP), TS 23.078, Sep. 2007. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23078.htm

[14] 3GPP, "Functional stage 2 description of Location Services (LCS)," 3rd Generation Partnership Project (3GPP), TS 23.271, Sep. 2007. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23271.htm

[15] "Open cellid." [Online]. Available: http://opencellid.org/

[16] "Shodan: The search engine for internet of things." [Online]. Available: https://www.shodan.io/

[17] O. Coskun, "Why nation-station malware targets telco networks," 2015. [Online]. Available: http://www.slideshare.net/merCokun1/defcon23-why-nationstatemalwaretargettelcoomercoskun-51440112

[18] D. Fox, "Der imsi-catcher," *Datenschutz und Datensicherheit*, vol. 26, no. 4, pp. 212–215, 2002.

[19] 3GPP, "Diameter based protocols to support Short Message Service (SMS) capable Mobile Management Entities (MMEs)," 3rd Generation Partnership Project (3GPP), TS 29.338.

[20] 3GPP, "Basic call handling; Technical realization," 3rd Generation Partnership Project (3GPP), TS 23.018, Sep. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23018.htm

[21] 3GPP, "MME Related Interfaces Based on Diameter Protocol," 3rd Generation Partnership Project (3GPP), TS 29.272, Sep. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/29272.htm

[22] C. Bonnet, "From ss7 to diameter security," 2015. [Online]. Available: http://www.slideshare.net/zahidtg/from-ss7-to-diameter-security

Representation Selection Problem: Optimizing Video Delivery through Caching

Andrea Araldo*†, Fabio Martignon*§ and Dario Rossi†
* LRI, Université Paris-Sud, {first.last}@lri.fr
† Telecom ParisTech, {first.last}@enst.fr § IUF, Institut Universitaire de France

***Abstract*—To cope with Internet video explosion, recent work proposes to deploy caches to absorb part of the traffic related to popular videos. Nonetheless, caching literature has mainly focused on network-centric metrics, while the quality of users' video streaming experience should be the key performance index to optimize. Additionally, the general assumption is that each user request can be satisfied by a single object, which does not hold when multiple representations at different quality levels are available for the same video.**

Our contribution in this paper is to extend the classic *object placement* problem (which object to cache and where) by further considering the *representation selection* problem (i.e., which quality representation to cache), employing two methodologies to tackle this challenge. First, we employ a Mixed Integer Linear Programming (MILP) formulation to obtain the centralized optimal solution, as well as bounds to natural policies that are readily obtained as additional constraints of the MILP. Second, from the structure of the optimal solution, we learn guidelines that assist the design of distributed caching strategies: namely, we devise a simple yet effective distributed strategy that incrementally improves the quality of cached objects. Via simulation over large scale scenarios comprising up to hundred nodes and hundred million objects, we show our proposal to be effective in balancing user perceived utility vs bandwidth usage.

***Keywords*—Content Distribution; Optimization; Quality of Experience (QoE); Caching**

I. Introduction

The large majority of the Internet traffic currently consists of video delivery. The related traffic is expected to explode due to increasing demand on the one hand, but, more importantly, in reason of the increasing quality expectations of users. Indeed, at the Consumer Electronics Show in Las Vegas, "Beyond 4K Ultra HD" technologies were shown that increase pixel density by 167% [1] over the previous year – a much faster growth rate with respect to worldwide user population.

Caching video content, with either current Content Distribution Network (CDN) technologies and their interconnection or more futuristic and pervasive Information Centric Network (ICN) architectures, may help containing this traffic deluge. However, the caching literature has, with few exceptions [3], considered network-centric metrics like hit-ratio, hit-distance, server offload, etc., overlooking more important aspects related to the quality of user experience.

More importantly, except for some recent effort, video streaming and caching have been mostly studied as orthogonal problems, often in different research communities. Rephrasing the title of [12], caching and video are still not friends: classic video streaming mechanisms assume that a client downloads a video from a single source, which is not true in presence of caching, misleading control loops. Moreover, caching techniques are designed with generic content in mind, whereas we show in this paper that the peculiarities of video traffic demand for caching mechanisms specifically tailored for it. The most important peculiarity is a different request-to-object mapping assumption: previous studies assume that a user request can be mapped to a single object, while a request for a video can be served by providing one of the different representations of the same video, corresponding to different quality levels, and ultimately different levels of user satisfaction.

As a first consequence, it is no longer sufficient to choose which *object* to cache, but also which of its available *representations*. Therefore, we add a new dimension to caching techniques: in addition to the classic *object placement* problem, i.e., which object to cache and where, we also consider the *representation selection* problem, i.e. which quality representation to cache. As a second consequence, the bandwidth required to satisfy a certain request is no more univocally determined by the object identifier, but depends on the quality at which we decide to serve that request. ISPs can leverage the possibility of serving the same request by using different bandwidth amounts to efficiently exploit their links and adapt to the dynamics of traffic, maximizing user satisfaction at the same time.

We consider a scenario in which Autonomous Systems (AS) peer together forming a coalition to collaboratively share their cache resources. We do not investigate the coalition formation problem, and rather focus on providing a strategy for the AS coalition to maximize the quality perceived by their users. Our key contributions can be summarized as follows:

- We propose a novel representation-aware Mixed Integer Linear Programming (MILP) model, which determines the object placement, quality representation selection and routing, taking into account video quality in order to maximize users' experience, in a capacity and cache size-constrained network scenario.
- The knowledge gained by studying the structure of the optimal solution inspires the design of a distributed caching strategy, which we implement in an event-driven simulator to scale up the analysis to network sizes of hundred nodes and catalog of hundred million objects.

Our key finding is that, despite the cache deployment considerably helps in improving user quality of experience, utility maximization can be achieved by (i) minimizing the number of representations stored per object (to increase the cache

ISBN 978-3-901882-83-8 © 2016 IFIP

efficiency), and (ii) selecting the most useful representation for each object (which is at the heart of the representation selection problem). We thus devise a simple yet effective distributed strategy that: (i) maintains a single representation per object, and (ii) incrementally improves the quality of cached objects at each new request, so that the average quality in steady state is inversely related to the object popularity.

This paper is structured as follows. Sec. II casts this work in the context of related effort. Sec. III introduces the methodologies used in our work, extensively describing (i) the representation-aware MILP model and its variants, as well as (ii) the online distributed cache algorithm. Sec. IV illustrates our numerical and simulation results in both (i) toy case scenarios, to understand properties of the optimal solution as well as (ii) large scale scenarios, to confirm our reasoning to hold in more general cases. Sec. V concludes the paper with a summary of our findings and our future work agenda.

II. Related work

Video streaming over the Internet has become a mainstream research topic in recent years: as such, several works focused on the problem of ensuring an efficient *video streaming* in communication networks. Similarly, *caching* is a very effective technique that permits to serve contents in both bandwidth and time-efficient manners, which has attracted a surge of attention in recent years through popularization of Content Distribution and Information Centric networks. However, as already discussed, there is still lack of a unified viewpoint to alleviate the huge increase in required bandwidth and guarantee satisfactory Quality of Experience (QoE) for users.

To confirm this, classic caching directly applied to video streaming is not only inefficient but can even be harmful [12]. Another example of classic caching vs. classic video streaming impairment is given in [15], which, by means of trace-driven simulations, finds that an ICN cache deployment would not lead to relevant QoE improvement in video delivery. Yet we argue that such results understate the benefits achievable via caching, since they are obtained by applying representation-blind policies, which consider homogeneous objects, all encoded at a single quality. In this work, we instead leverage the possibility to serve different quality representations to maximize user satisfaction, respecting capacity constraints.

Conversely, QoE maximization has been tackled in the classic video literature [6], [12], [13] by proposing control mechanisms that intelligently share bandwidth among different users. Control algorithms in scenarios with multiple sources (like caches and repositories) are proposed by [12], [13]. In particular, the former shows that quality fluctuations can be observed because of caching, which hampers QoE. Both works evaluate control algorithms under a given content allocation, whereas we look for the allocation guaranteeing the best QoE. Authors of [14] consider caching of videos in a heterogeneous network, assuming that users can specify the minimum video quality they are willing to accept and the network provider goal is to minimize delay and cost while providing at least that quality. Our viewpoint is different, since we directly measure user satisfaction in terms of quality provided, rather than delay, and our goal is not just to satisfy a minimum requirement but to send videos at the maximum possible quality. In a similar context, the work in [19] introduces a new layered video encoding, while our enhancement is obtained using the currently most deployed technologies, like MPEG-DASH. Moreover, the context of our model is a multi-AS environment, where the capacity of multi-hop paths limits the rate of transmission (thus, the served quality) whereas in wireless contexts the limitation is due to the channel condition. The closest work to ours is perhaps [10], which employs caching, transcoding and routing functions to minimize the networking cost in a video distribution context. A two-step iterative approach is proposed, where, first, storage and computing resources are allocated optimally, then the routing is configured in the second phase. However, the model does not explicitly account for the utility perceived by users downloading different video representations, which is the focus of our paper.

The fact that a single video can be represented at different qualities has an important impact on users' experience, which [17] and [20] study in a CDN and wireless scenario, respectively, investigating what is the subset of video quality levels to make available in order to maximize QoE. Both make crude simplifications of the network settings: the former characterizes a delivery system only by the total bandwidth, while the latter only considers one cache and one video. Differently, we assume that the set of quality levels is already established, and look at the problem from a network viewpoint.

III. Methodology

This section explains our methodologies, casting them to an AS-level system model (Sec.III-A). We formulate a Mixed Integer Linear Programming (MILP) model that maximizes the users' quality of experience, taking into accurate account the different object representations available, as well as capacity and cache constraints (Sec. III-B), discussing its limits and possible extensions (Sec. III-C). We constrain our model to give solutions with simpler structures guaranteeing, at the same time, performance close to the optimum (Sec. III-D). The solution of the MILP, that we report in a later section, then guides the design of an effective distributed caching policy that can be easily implemented in practice (Sec. III-E). Tab. I summarizes the notation used throughout this paper.

A. System model

We illustrate the system model considered in our work, with the help of an example scenario depicted in Fig. 1. We consider a set $V = \{1, \dots, V\}$ of Autonomous Systems (ASes), whose interconnection is represented by a graph, composed of nodes and capacitated arcs. Nodes in the graph (ASes) can act as *content producers* (when they are directly connected to some repositories), *transit* ASes that merely participate in the content caching and diffusion, or *consumer* ASes that additionally generate video requests. Repositories and caches distributed in the ASes store objects (in particular, multimedia content and videos), of which different representations (quality levels) exist, belonging to a discrete set Q. Each of these quality levels is associated to a rate r^q necessary to support and transmit the object at the given quality q, as well as to a storage space s^q that is necessary to cache it. AS users issue requests for videos without specifying the quality representation, given that the model will find the optimal one.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Table I. SUMMARY OF THE NOTATION USED IN THIS PAPER.

Parameters of the Models	
A	Set of arcs
V	Set of Nodes (Autonomous Systems, ASs)
O	Set of objects
Q	Set of qualities
$FS(i)$	Set of forward arcs $(i,j) \in A$ for node $i \in V$
$BS(i)$	Set of backward arcs $(j,i) \in A$ for node $i \in V$
b_e	Capacity of the arc $e \in A$
n_v^o	Number of requests for object o, in AS $v \in V$
r^q	Rate required to retrieve an object at quality $q \in Q$
s^q	Storage space required to cache an object at quality $q \in Q$
U^q	Utility gained to serve one request for an object at quality q
p_v^o	0-1 Producers reachability matrix $p_v^o = 1$ if AS v has a producer for object $o \in O$ (it can serve whatever quality of object o)
S_v	Max caching storage that can be installed at AS v
S_{TOT}	Max caching storage that can be installed in the network
bw_v	Max egress capacity for AS $v \in V$, $bw_v = \max\left(\sum_{e \in FS(v)} b_e; \sum_{o \in O} n_v^o \cdot \max_{q \in Q} r^q\right)$

Decision Variables of the Models	
$n_v^{o,q}$	Number of requests for object o at quality q satisfied at AS v
$x_{v_s}^{o,q}$	0-1 Caching variable, if the source AS $v_s \in V$ caches o at quality q
y_e^{o,q,v_d}	Flow on arc $e \in A$ for object $o \in O$, at quality q sent to the destination AS $v_d \in V$
d^{o,q,v_d}	Rate requested at AS $v_d \in V$, for object o at quality q
$z_{v_s}^{o,q,v_d}$	Rate provided by the source AS $v_s \in V$, for object o, at quality q for the destination $v_d \in V$, when v_s behaves as a producer ($p_{v_s}^{o,q} = 1$)
$w_{v_s}^{o,q,v_d}$	Rate provided by the source AS $v_s \in V$, for object o, at quality q for the destination $v_d \in V$, when v_s behaves as a cache ($x_{v_s}^{o,q} = 1$)

Each AS has *upstream links* through which data is retrieved from other nodes and *downstream links* through which data is sent to users. ASes are endowed with caching capabilities, and can store objects as well as route object requests/data towards neighbor routers, the repository or clients. To be as general as possible, we do not specify the details of the technology that provides caching capabilities (ICN, CDN, Web proxy, etc.). Each object can be served at different qualities, which may depend on the network characteristics (link capacities, bottlenecks) and the clients position, and produce a utility that is experienced by users. The aim of our work is to determine (i) optimal allocations of objects to AS caches, (ii) optimal quality level(s) to store for each cached object and to map to each request, as well as (iii) optimal routing strategies, that collectively contribute in maximizing the overall utility perceived by network users.

B. Representation-Aware MILP

The Representation-Aware model that maximizes users' utility can be formalized as follows:

$$\max \sum_{o \in O} \sum_{q \in Q} \sum_{v \in V} n_v^{o,q} U^q \tag{1}$$

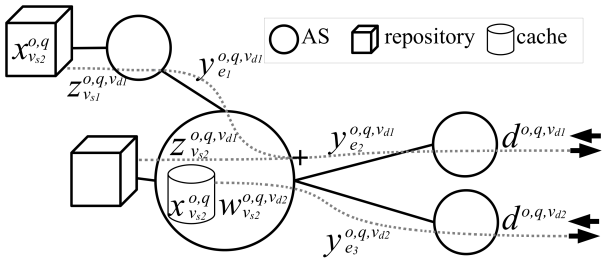


Figure 1. Example scenario and the main variables of the MILP model.

subject to:

$$\sum_{q \in Q} n_v^{o,q} = n_v^o \qquad \forall o \in O, v \in V \tag{2}$$

$$d^{o,q,v_d} = n_{v_d}^{o,q} \cdot r^q \qquad \forall o \in O, q \in Q, v_d \in V \tag{3}$$

$$d^{o,q,v_d} = z_{v_d}^{o,q,v_d} + w_{v_d}^{o,q,v_d} + \sum_{e \in BS(v_d)} y_e^{o,q,v_d} - \sum_{e \in FS(v_d)} y_e^{o,q,v_d} \qquad \forall o \in O, q \in Q, v_d \in V \tag{4}$$

$$z_{v_s}^{o,q,v_d} + w_{v_s}^{o,q,v_d} + \sum_{e \in BS(v_s)} y_e^{o,q,v_d} = \sum_{e \in FS(v_s)} y_e^{o,q,v_d} \qquad \forall o \in O, q \in Q, v_s \in V, v_d \in V, v_s \neq v_d \tag{5}$$

$$\sum_{o \in O} \sum_{q \in Q} \sum_{v_d \in V} y_e^{o,q,v_d} \leq b_e \qquad \forall e \in A \tag{6}$$

$$\sum_{v_d \in V} z_{v_s}^{o,q,v_d} \leq p_{v_s}^{o,q} \cdot bw_{v_s} \qquad \forall o \in O, q \in Q, v_s \in V \tag{7}$$

$$\sum_{v_d \in V} w_{v_s}^{o,q,v_d} \leq x_{v_s}^{o,q} \cdot bw_{v_s} \qquad \forall o \in O, q \in Q, v_s \in V \tag{8}$$

$$\sum_{o \in O} \sum_{q \in Q} x_{v_s}^{o,q} \cdot s^q \leq S_{v_s} \qquad \forall v_s \in V \tag{9}$$

$$\sum_{o \in O} \sum_{q \in Q} \sum_{v_s \in V} x_{v_s}^{o,q} \cdot s^q \leq S_{TOT} \tag{10}$$

$$x_v^{o,q} \in \{0,1\} \qquad \forall o \in O, q \in Q, v \in V \tag{11}$$

$$n_v^{o,q} \in \mathbb{Z}^+ \qquad \forall o \in O, q \in Q, v \in V \tag{12}$$

$$y_e^{o,q,v_d} \in \mathbb{R}^+ \qquad \forall o \in O, q \in Q, v_d \in V, e \in A \tag{13}$$

$$d^{o,q,v_d} \in \mathbb{R}^+ \qquad \forall o \in O, q \in Q, v_d \in V \tag{14}$$

$$z_{v_s}^{o,q,v_d}, w_{v_s}^{o,q,v_d} \in \mathbb{R}^+ \qquad \forall o \in O, q \in Q, v_d \in V, v_s \in V. \tag{15}$$

In particular, objective function (1) represents the overall utility experienced by network users, which is maximized by our model. The set of constraints (2) makes sure that all the requests are served at one (or more) quality level(s). In the problem instances we add a "special" quality level $q = 0$, which represents unserved traffic demands: when serving quality $q = 0$, no bandwidth is required ($r^q = 0$); moreover, no utility is generated, $U^0 = 0$. Constraints (3)

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

set the value of the rate requested at AS a, for object o, at quality q. Such demand is satisfied in (4). In particular, it can be satisfied because: (i) the AS is a producer for that object (i.e.: $z_{v_d}^{o,q,v_d} = d^{o,q,v_d}$), (ii) the AS caches the object (i.e.: $w_{v_d}^{o,q,v_d} = d^{o,q,v_d}$), or (iii) the AS retrieves the object (i.e.: the sum of flows on incoming links).

Flow balance constraints are imposed in (5) and we bound the arc capacity in (6). Similarly, in (7) and (8), we limit the maximum emitted flows the AS sends when it behaves as a producer and a cache, respectively. The overall caching storage that can be deployed by an AS is bounded in (9), and we extend the same limit to the entire topology in (10). Finally, integrality and non-negativity constraints are imposed in (11)-(15).

C. Discussion

The problem that our model aims to solve can be conceptually formulated as follows: at a generic time instant we have facilities, i.e. link capacities and caches and a set of concurrent user requests for video chunks. Our goal is to find the facility allocation that maximizes users utility. In other words, we adopt a snapshot approach, as usually done in optimization works, which is based on this instantaneous picture of the system. Although this might be considered too simplistic, almost all the vast and notable literature, e.g. [5], [8], [10], [14], [17], [19], [20], which applies optimization models to network analysis is based on it, even when not explicitly stated, and results have been widely accepted by the community. For these reasons, the plausibility of the snapshot approach is unlikely to be questionable and, however, we build on it only to show meaningful insights on the novel representation selection problem, rather than to provide absolute measures. On the other hand, we analyze realistic scenarios, where requests can arrive at any moment and the system evolves from time to time, in Sec. IV-E by means of simulation.

While related work usually aims to minimize delay to improve user perception, we focus instead on maximizing the provided quality for two reasons: i) we want our contribution to be complementary to this related work, ii) the packet delay can be absorbed by playout buffers and be invisible to users. The only exception to this is when this delay is excessively high or variable, causing high startup times or rebuffering episodes. This happens in case of congestion. For these reasons, rather than looking at the delay, we focus on caching content at the right quality, such that it can be transmitted using the available bandwidth on the path, thus avoiding congestion.

Another aspect worth underlining is that in today's video delivery, plugins in the user Web browser select the quality representation to request, while we assume that ISPs choose the best possible quality to serve its users. This is not unrealistic since, in either case, users do not make any explicit choice most of the time [9], so that the selection mechanism, be it done in the Web browser of their personal device, or at the proxy in the ISP premises, is completely transparent to them.

Additionally, we remark that, while most of the effort in video streaming literature and industry is devoted to congestion control algorithms, which are outside the scope of this work, we orthogonally focus on caching and aim at finding the performance bounds from a more abstract viewpoint. The findings we provide here should be considered what an optimal caching strategy can theoretically achieve, supposing a perfect congestion control mechanism at the bottom. For this reason, we can adopt the snapshot approach.

Table II. MODEL VARIANTS IMPLEMENTING NATURAL AS POLICIES: ADDITIONAL CONSTRAINTS FOR THE MILP MODEL (1)-(15)

Caching Policy	Additional constraint in MILP model
NoCache	$x_v^{o,q} = 0$
CacheLQ	$x_v^{o,q} = 0, \forall q \neq LQ$
CacheHQ	$x_v^{o,q} = 0, \forall q \neq HQ$
AllQ	$x_v^{o,q_h} = x_v^{o,q_k}, \forall q_h, q_k \in Q$
Partitioned	$\sum_{o \in O} x_{v_s}^{o,q} \cdot s^q \leq S_{v_s} \cdot \frac{s^q}{\sum_{q' \in Q} s^{q'}}$

It is worth noticing that our model can be easily extended to have a fine-grained representation, considering heterogeneity of video type and user device. As for the former, it is known that videos with different subjects (sport, movies, TV shows), even if encoded at the same bit-rate and resolution, are perceived in a different way [17]. As for the latter, a user watching a video on a smartphone may be perfectly satisfied with a resolution and a bit-rate lower than the one demanded by a user using a ultra-HDTV 4K screen. However, this level of detail is beyond the scope of this paper and such directions can be incorporated at a later step in the model.

D. Modeling AS policies

Jointly deciding the optimal representations that each node should cache and serve to users is a hard task to be performed by a distributed online strategy, in which each node makes local decisions without having knowledge of the status of the rest of the network and the overall set of requests. Nonetheless, our final aim is to give a feasible solution that can be deployed in a real network, providing a good performance at least close to the optimal one.

We thus constrain our model to give solutions with a simpler structure and we verify how far they are from the optimum. The constrained variants of the model, detailed hereafter, are easier to approximate in distributed, online algorithms and, as our numerical results will show, some of them exhibit indeed very good performance, close to optimality in several situations. Thanks to the flexibility of our MILP model, modeling AS policies is as simple as adding a single constraint for each strategy.

Such constraints, specified in Tab. II, include: a *No Cache* strategy, which never caches videos; *CacheLQ* and *CacheHQ*, which exclusively cache the lowest (highest) quality representation available, indicated with quality level *LQ* (*HQ*), respectively; *AllQ*, which caches all quality representations for any cached object; finally, *Partitioned* stores the same *number* of objects for each quality representation (while their buffer occupancy depends on the quality of the corresponding representation). Note that the constraints only concern caching, and do not force to serve a request with a specified quality representation. For example, a HQ video can still be served, even when *CacheLQ* is employed; in such case, given that HQ videos cannot be cached, they must be retrieved directly by a repository and cross all links between this latter and the user. In this work, we assume that all ASes in the coalition use the same policy, chosen among the ones described above.

E. Distributed policy: Bandwidth-utility trade-off

In the MILP model, our only objective is to maximize the utility. To do so, we let the model use the links at their full capacity. In practice, ASes may tend to limit link utilization, in order to avoid the occurrence of congestion, to ensure low end-to-end latency and bound operational costs associated to traffic transmitted toward transit providers. Therefore, a trade-off arises between the utility provided to users and the bandwidth used to guarantee it, that we investigate via simulation.

The bounds of this trade-off are represented by two scenarios: namely *OnlyLQ* (*OnlyHQ*) where objects have a single representation equal to the lowest (highest) quality level. These policies respectively correspond to a crude attempt to minimize the bandwidth (vs maximize the utility) but, as a consequence, incur in low user utility (vs high bandwidth usage).

Between these two extremes, we propose a *Quality Improvement* (*QImpr*) strategy that reactively incrementally improves quality of stored replicas at each new request, and opportunely balances bandwidth and user utility. *QImpr* operates as follows. Each request $req(q)$ carries a value q specifying the minimum required quality, which is always set to $q = 1$ at the ingress of the network (either by the user browser or the ISP proxy) meaning that any quality is accepted (i.e., receiving a LQ representation is preferable to not receiving the video at all). When a new request $req(q)$ arrives at any cache, if a copy at quality $q_{\text{cached}} \geq q$ is found, the request is served with that copy and, at the same time, the AS node issues another request $req(q_{\text{cached}}+1)$ for the same object. Otherwise, $req(q)$ is normally forwarded.

Caches maintain objects in an ordered list. Whenever an object o at quality q arrives, if a better quality $q_{\text{cached}} \geq q$ of that object is already cached, the incoming object is discarded. Otherwise, the new object representation (i) is placed at the head of the list, (ii) any lower quality representation of o is evicted, (iii) if further space is needed to store o, this is obtained by evicting cached objects starting from the least recently used one, up toward the head, until a sufficient space to accommodate o at quality q is available. Shortly, expected benefits of this policy are that unpopular objects will only be cached at low quality, whereas popular objects will quickly escalate quality levels. On the downsides, popular objects will be requested at multiple quality levels, generating a slight overhead in the quality improvement process.

Note that in reason of size heterogeneity between representations at different levels, caching a new object causes the eviction of a variable number of least-recently-used objects sufficient to make room for the incoming higher quality representation. This is in contrast with what usually assumed in the ICN-flavored caching literature that assumes all chunks having equal size.

Table III. QUALITY LEVELS AND CORRESPONDING TRANSMISSION RATES, CACHE OCCUPANCY AND PERCEIVED UTILITY (LINEAR/CONCAVE).

Quality	Rate (Kbps)	Utility $u_1(q)$	Utility $u_2(q)$
1	300	0.2	0.67
2	700	0.4	0.80
3	1500	0.6	0.88
4	2500	0.8	0.95
5	3500	1	1

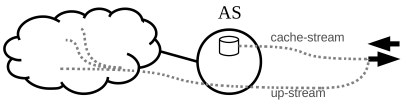


Figure 3. Single-AS scenario: Videos can be downloaded from the cache (cache-stream) or from other ASes (up-stream).

IV. RESULTS

This section evaluates the impact of caching on the overall video quality perceived by users, showing the validity of the caching strategies proposed so far. To this aim, we provide both numerical solutions of the MILP model via the CPLEX 12.5 solver for the centralized policies and the results of discrete event simulation, implemented in ccnSim [2], for the distributed solutions. After describing the scenario in Sec. IV-A, we investigate performance and properties of the proposed strategies in an incremental fashion.

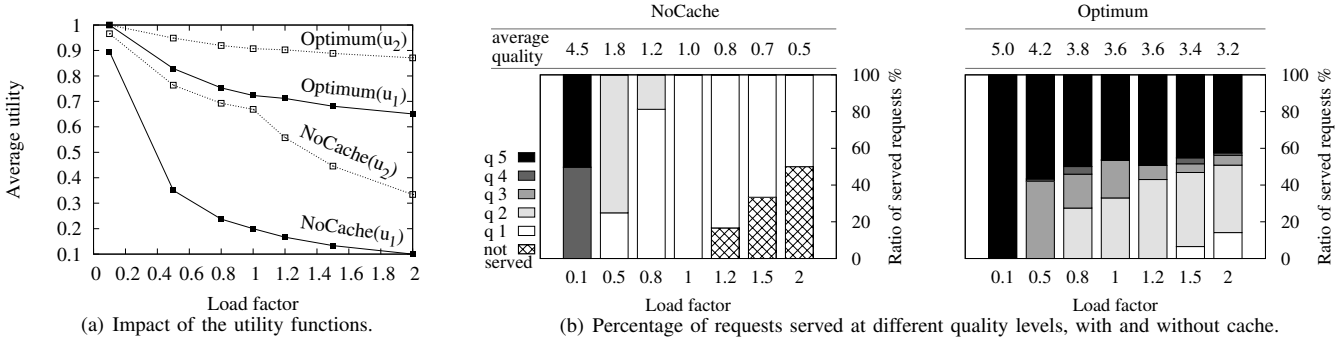
Focusing on a single AS, we first illustrate the structure of the optimal solution in Sec. IV-B. We then thoroughly analyze the bandwidth-storage tradeoff in light of variable representations in Sec. IV-C. We next contrast the range of centralized policies, as well as the distributed *QImpr* policy, in Sec. IV-D. Moving to a multi-AS scenario, we finally confirm MILP results to hold on a 10-node topology, and extend the simulation results to cover topologies up to 100-nodes and catalogs up to 10^8 objects in Sec. IV-E

A. Scenario

We consider five quality levels [7] in the set Q as reported in Tab. III. Each quality corresponds to a given resolution and bitrate, which both increase for increasing quality levels. We only report the bitrate as this is more pertinent to our optimization goal: video bitrate correlates to both cache storage space, as well as network bandwidth. Resolution, instead, does not come into play directly in the system model, apart from determining a different user perception, that is accounted for in the utility function.

The utility function must be an increasing function of the provided quality, since the higher the quality provided to a user is, the better the utility. Moreover, it must be concave to express the diminishing return in the experience of human vision when providing improved quality [16]. The exact shape of such an utility function is still subject to debate, and there is no unanimously accepted function. However, gathering this function is a hard task that requires intensive experimentation with real users, which is far from the topic of this work. To gather results that are not tied to a specific function, we consider two shapes at the broad end of the spectrum of plausible utility functions, tabulated in Tab. III. Specifically, we define $u_1(q)$ as a model with linear return with respect to the quality: the model likely underestimates contributions of low quality videos, and does not exhibit diminishing returns [16], so that it is biased toward high-quality content. We next define $u_2(q)$ as a power function with a higher concavity: this model does exhibit diminishing returns but sits at the other side of the spectrum as it possibly overestimates contributions of low-quality videos (notice indeed that $u_2(q = 1) > 3u_1(q = 1)$). In the following, we will report the *average system utility* as

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



(a) Impact of the utility functions.

(b) Percentage of requests served at different quality levels, with and without cache.

Figure 2. Single-AS scenario: (a) Benefits of optimal caching and impact of utility function; (b) breakdown of the utility across quality levels for varying load.

the average per-request utility, i.e. the total utility as in (1) divided by the total number of requests.

Unless otherwise stated, we consider the single AS scenario depicted in Fig. 3: at a logical level, in the cache-stream we include the flows of videos downloaded from the cache, while up-stream includes the flows retrieved through other ASes. The cache represents the aggregate of several cache nodes within the AS, and similarly the up-stream resource represents a single logical link, aggregating all physical links where the request can be satisfied (i.e., all the links except the one where the request is coming from).

Also, unless otherwise stated, the catalog comprises $O = 10^4$ objects whose popularity is distributed as a Zipf with exponent $\alpha = 1$. The cache space at each AS is sufficient to store 1/100 of the catalog objects at the highest quality, *HQ*. Observe that the size of each object depends on its quality, i.e. an object at quality q is s^{HQ}/s^q times smaller than an object at the highest quality, with s^{HQ}/s^{LQ} exceeding one order of magnitude as can be seen in Tab. III.

All links have the same capacity b. We express the number of user requests as a *load factor* L, i.e. the factor by which we should multiply b in order to transmit all the requested objects at the lowest quality, *LQ*. Otherwise stated, if the load factor is $L = 1$, even if no cache is deployed in the network, we can satisfy all requests at quality LQ, by fully utilizing the network capacity. Notice, however, that due to cache space, it makes sense to consider a normalized load larger than $L > 1$, since part of the endogenous requests can be served from the cache without consuming upstream bandwidth.

B. Structure of the optimal solution

We first start assessing the dependency of our results on the particular perceptual model in the single cache scenario, based on the results from the MILP model. We contrast two extremes, namely the optimal solution against the case in which the system is not equipped with caches (so that this latter can sustain a load at most equal to $L = 1$). The average utility is shown in Fig. 2-(a) for both linear $u_1(q)$ and concave models $u_2(q)$: while quantitative results are of course affected by the peculiar function, qualitative results are instead independent of the utility function considered. In particular, the improvement of user experience provided by optimal caching is notable at high load, where caches at the AS absorb a large fraction of the requests, alleviating the impact of the upstream bandwidth limitation. Since the qualitative results between $u_1(q)$ and $u_2(q)$ remain unchanged, and to avoid cluttering the pictures, we only consider the concave profile of $u_2(q)$ in what follows.

A breakdown of the quality levels served to users is reported in Fig. 2-(b), which helps to better understand the structure of the optimal solution – thus, ultimately, where the utility gain comes from. Without any cache, all the delivered videos must cross the upstream link, and the bandwidth is hardly available to transmit them at high quality, unless the input load is particularly low ($L = 1/10$). At high load, the bandwidth is not sufficient to serve all the requests, not even at the lowest quality, and a growing fraction remains unsatisfied. The situation is drastically improved by optimal caching, which stores a significant fraction of videos, and *especially the most popular ones, at high quality*. Since the requests for these videos account for a large part of the overall requests, the upstream link is relieved of a considerable amount of traffic. As a first consequence, we are able to satisfy all the requests coming from users. Moreover, the most popular objects are served at high quality, which as net effect increases the average utility perceived by users.

C. Contributions of cache and upstream link

To better understand the relative contribution of storage vs. bandwidth, we decompose the video flows arriving at users in *cache-stream* and *up-stream*, where the former is the stream of data retrieved from the cache, whereas the latter is the flow coming from the upstream links, as illustrated in Fig. 3. Based on the results from the MILP model, we represent the breakdown of the utility provided by content retrieved from cache vs. content retrieved from upstream in Fig. 4, where the sizes of the circles represent the relative contribution of the two utility values. Circles additionally report the quality breakdown of the two contributions.

From Fig. 4 we first observe that, in the scenarios under consideration, the cache is responsible of the most part of the utility (storage circles are bigger than upstream ones), as it stores the most popular objects (thus intercepting a large fraction of traffic) at a furthermore high quality.

Second, an interesting specialization arises between the cache-stream and up-stream: the highest quality levels (darker colors) are served by the cache and only low representations cross the upstream link. Indeed, it would not be beneficial to serve high quality objects through the upstream link, since the

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

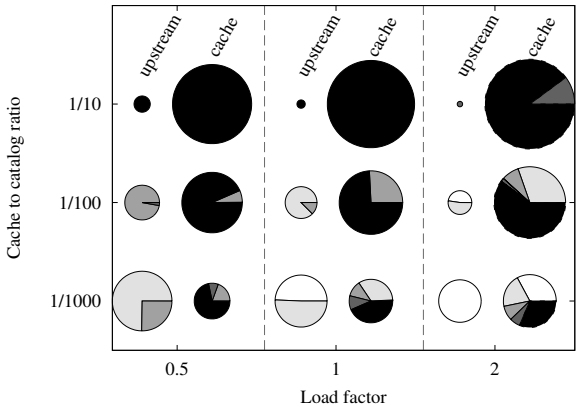


Figure 4. Single-AS scenario: Contributions of cache and upstream links. Circle size reflects relative cache vs upstream contribution, and the breakdown reports the qualities of both contributions.

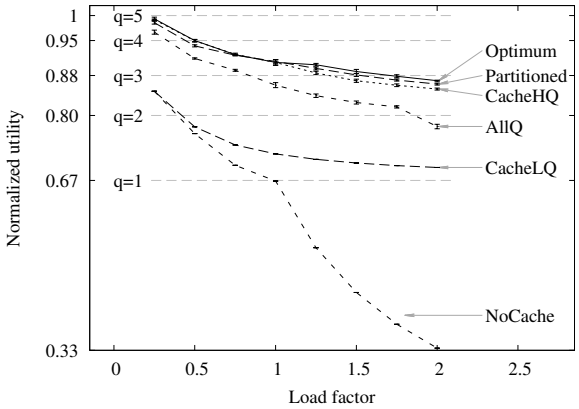


Figure 5. Single-AS scenario: comparison of MILP variants.

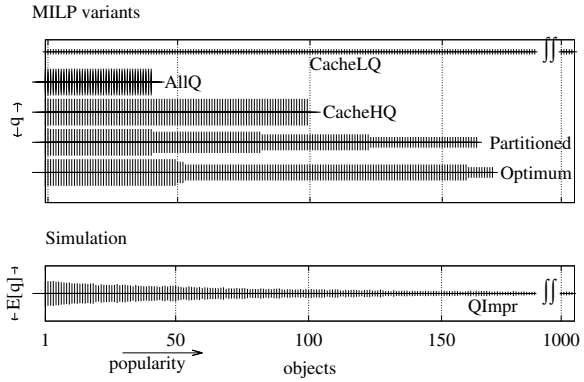


Figure 6. Single-AS scenario: Quality levels cached by centralized strategies (MILP variants) and distributed policy (simulation).

high bandwidth cost should be paid repeatedly, at each request. On the contrary, placing them in the cache permits to pay only once the cost in terms of memory, and to still repeatedly gather utility at each request.

Third, the load has an evident impact on the breakdown. At high load, both streams must carry lower quality representations. Indeed, in this case, the average quality of the upstream must be low, to fit the link capacity. At the same time, we need to reduce the number of transmissions on the upstream by intercepting more requests with cached copies. To do so, we need to cache a larger number of different videos and, since the cache space is limited, we need to store smaller copies of them, i.e. lower quality representations. This explains why, at high load, the quality of the cache-stream decreases.

Fourth, we observe the impact of cache size on the breakdown: as expected, when the cache size increases, its relative contribution to the overall utility increases as well. Yet, more interestingly, also the breakdown of the stored video quality changes as well: in particular, the larger the cache, the higher the quality, which is intuitive.

Finally, observe that the cache size has a side effect on the breakdown of the upstream video quality: indeed, the average quality increases for increasing cache size, which can be explained with the fact that the larger the cache, the larger the fraction of absorbed traffic. As a consequence, at any given load the upstream link has to serve less requests and can afford to do it at higher quality.

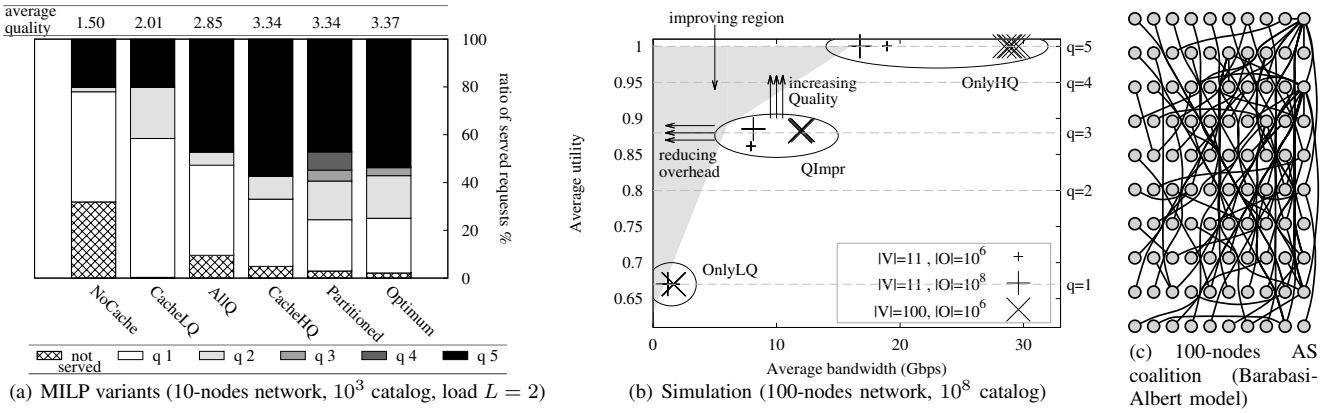
D. Performance bounds of online algorithms

We next compare the performance of the five strategies discussed in Sec. III-D (viz., *NoCache, CacheLQ, CacheHQ, AllQ, Partitioned*), with the solution that maximizes the quality of experience perceived by network users (*Optimum*). Utility is reported in Fig. 5, whereas the structure of the solution is reported in Fig. 6, which depicts the quality level of each stored object under all strategies – including the distributed *QImpr* policy discussed in Sec. III-E.

Note that, when the network caches only low quality objects (*CacheLQ*), their small size permits to store a large number of them, intercepting a large fraction of the requests. This already provides robustness with respect to load, guaranteeing at least a minimum quality (the *CacheLQ* curve is above the $q = 1$ reference quality), that is not possible without cache. However, *CacheLQ* does not exhibit cache efficiency because higher quality objects, necessary to increase the average utility, can only be retrieved through the upstream link. Rigidly storing all quality representations (*AllQ*) further improves the performance but is still far from the optimum. Indeed, for each object we must waste cache space for all the representations, although only a subset of them will be actually served to users. This limits the number of different objects that can be cached. *CacheHQ* performance approaches to the *Optimum*, suggesting that storing few (due to their large size) popular objects at HQ already provides a notable payoff (due to the product of their popularity times their utility at HQ).

Yet, *Partitioned* performance is even closer to the *Optimum*: the root cause is that the quality representation selection is similar to the optimal one, as Fig. 6 shows. In particular, the optimal behavior in terms of overall utility is to store a number of objects at each quality, preferring to store more popular objects at higher quality, and *Partitioned* implements this behavior. This increases the overall cardinality of cached content and assigns to each object the "right" quality, i.e. the one such that the cost in terms of occupied memory is

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



(a) MILP variants (10-nodes network, 10^3 catalog, load $L = 2$)

(b) Simulation (100-nodes network, 10^8 catalog)

(c) 100-nodes AS coalition (Barabasi-Albert model)

Figure 7. Multi-AS scenarios: (a) optimal solution, (b) simulation results and (c) example large-scale topology.

compensated by the pay-off in terms of utility provided to the set of requests for it. The difference between *Partitioned* and *Optimum* is in the number of objects stored at each quality. While *Partitioned* constrains this number to be the same for each quality, *Optimum* does not incur this constraint and prefers, in this scenario, roughly two quality levels. So doing, the *Optimum* strategy caches more objects than *CacheHQ* (but less than *Partitioned*), a significant fraction of which is at lower quality than *CacheHQ* (but higher than *Partitioned*).

From the above observations, we infer that the quality at which each object must be cached should increase with its popularity. This is the observation we leveraged in the design of *QImpr*, which is shown in the last raw of Fig. 6. Notice that while solving the optimization problem returns exactly one object quality, in the simulation case the representation of an object stored in the cache varies over time, so that we report the average quality for an object sampled at 100 random times during the simulation. It can be seen that *QImpr* tends to store only the popular objects at high quality, thus approaching a solution that is structurally similar to *Partitioned* or *Optimal* strategies – which confirms the mechanism of improving the object quality at each new hit to pay off. Note that a fairness concern may arise, since popular content is served better than the rest. In any case, bandwidth is limited and it is impossible to serve all the content at high quality. Therefore, a network provider has two choices: i) being fair and lowering the quality of all the served videos or ii) differentiating based on popularity. While the former case is admissible, we have shown that the latter permits utility maximization, which is the target of this paper. On the other hand, a network provider may wish to provide always a quality above a certain threshold higher than LQ. We can easily model this by removing from the set Q of the admissible levels the lowest ones.

E. Realistic and large scale topologies

We now consider a multi-AS environment, where each AS operates a cache system with a storage space sufficient to cache 1/100 of the catalog at the highest quality. We start our analysis solving the MILP model for a coalition of 10 ASes and a 10^3 objects catalog in Fig. 7-(a), and extend the analysis up to 100 ASes and 10^8 objects in Fig. 7-(b) The multi-AS graphs are generated in accordance to the Barabasi-Albert model [4], which is considered to approximate the AS interconnection in Internet [18]. A compact illustration of the interconnection is in Fig. 7-(c) for a large scale topology of 100 nodes.

We only briefly comment MILP results, reported in Fig. 7-(a), to assess that they are coherent with the observations on the single-AS case. Specifically, we notice that while the average quality is very similar among *CacheHQ, Partitioned* and *Optimum*, however the fraction of content that is not served is largely different. In the case of *CacheHQ*, about 5% of the videos are not served, which is 2.3 times larger than the fraction of non-served videos in the *Optimum* case. In contrast, the *Partitioned* strategy limits to +30% the amount of additional videos not served with respect to *Optimum*.

While this fact does not appear in the perceptual model we used (where a non served content has a utility 0 and does not generate any penalty) nevertheless it can be argued that the impact of service denial can be much worse. Indeed, from loss aversion models commonly used in prospect theory [11], not receiving a video at expected quality q generates a negative utility $-2u(q)$, which could be accounted for in the model. Yet, repeatedly receiving denial of requests could lead users to change ISPs on a long timescale, which can have disastrous consequences on the ISP business, for which limiting the fraction of non-served content is primordial.

A second important observation is that gains are structurally equivalent to what early shown in the single-AS case, and on which we based the design of our proposed distributed strategy (*QImpr*). Comforted by this observation, we relax the capacity constraint and carry out simulation of *QImpr* on large scale instances. Otherwise stated, while the *Optimum* operates at full capacity, we do not expect ASes to run their network at this capacity regime. Rather, ISPs will be interested in controlling their average (or peak) bandwidth on external links, which we assess by simulating scenarios with dynamic arrivals. Aiming at assessing the utility vs. bandwidth trade-off, we use two additional reference points where only low-quality (*OnlyLQ*), or high-quality objects exist in the system (*OnlyHQ*), and caches employ standard Least Recently Used (LRU) replacement. Requests are generated according to a Poisson process and results are collected in steady state over 10 runs for each scenario. The bandwidth utilization is computed considering that, every time an object at quality q crosses a

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

link, it occupies a bandwidth r^q. The bandwidth in Fig. 7-(b) is averaged over time and over all the links of the network.

Note that the points in Fig. 7-(b) are well clustered, meaning that the performance of *OnlyLQ*, *OnlyHQ* and *QImpr* is coherent and our findings do not vary with the scale of the problem. Bounds on the utility vs. bandwidth trade-off are given by *OnlyLQ* and *OnlyHQ*: the former guarantees the minimum bandwidth utilization by only serving and caching objects at the lowest quality, while the latter provides maximum utility at the expense of high bandwidth utilization. *QImpr* nicely fits halfway these extremes, realizing a smooth tradeoff between bandwidth and quality.

The picture finally shades an area where the performance of interesting distributed algorithms lays: i.e., those that achieve a more convenient bandwidth-quality tradeoff. *QImpr* design can be ameliorated to move performance in the upper-left part of Fig. 7-(b) by (i) reducing the overhead (i.e., move left) and (ii) improving the utility (i.e., move up). As far as (i) *overhead* is concerned, recall that whenever a request hits a cached copy at quality q, the cache immediately triggers a request to improve the content quality to $q+1$. These cache-originated requests constitute an overhead, which could be limited by probabilistically reducing the rate at which they are issued – much as in probabilistic meta-caching. As far as (ii) *utility* is concerned, recall that the *Optimal* solution implicitly quantized the quality levels to a subset of all the available ones, which should be easy to implement.

Notice however that overhead reduction and utility maximization are conflicting goals, since e.g., slowing down the rate at which quality of content is improved from q to $q+1$ by a given factor also implies that the amount of requests served at quality q instead of $q+1$ grows by the same factor. While this observation affects only the transient but vanishes in the steady state, it can however be argued that it has practical relevance in real scenarios where popularity is time-varying and there is no steady state. Additionally, the choice of the best subset may depend on the utility, cache/upstream ratio, topology, request load, popularity skew, etc. which requires future work.

V. Conclusions

To the best of our knowledge, this is the first paper that tackles the problem of optimal content distribution in cache-enabled networks, by explicitly taking into account multiple representations of the same object, each having a different utility perceived by users. This is a crucial aspect in video delivery, in which each video can be represented at different quality levels. The need for caching techniques that, apart from the general ones, are optimized for video traffic is enforced by the prevalence of this traffic on the other types and its inherent cacheability. We find the optimal caching solution that maximizes user utility and we contrast it against several candidate strategies along the user experience angle. We study the fundamental properties of the solution to infer important guidelines to optimize object-level caching in video delivery. We leverage these guidelines in designing a distributed solution that we benchmark via event-driven simulation. Our key findings suggest that (i) the quality at which each object should be cached is inversely related to its popularity, (ii) a balance between user perceived utility and bandwidth usage is possible by means of intelligent caching distributed policy of which *QImpr*, the one proposed in this paper, is an example.

However, *QImpr* does not allow ISPs to explicitly control the balance, so to reach a target network utilization. In our future work, we aim at (i) proposing a distributed solution that approaches a target optimal bandwidth-quality tradeoff, as well as (ii) performing a thorough sensitivity analysis on the topology of the coalition, investigating how cache content differentiates with respect to the node position in the network, due to the interaction and the filtering effect of neighbors.

Acknowledgment

This work was funded by Labex DigiCosme (project ANR-11-LABEX-0045-DIGICOSME) operated by ANR as part of the program Idex Paris-Saclay (ANR-11-IDEX-0003-02) and partly by ANR Green-Dyspan and carried out at LINCS http://www.lincs.fr.

References

[1] Sharp Website. http://www.sharpusa.com/ces-2015-recap.aspx.

[2] ccnSim. http://www.enst.fr/~drossi/ccnSim.

[3] M. Badov, A. Seetharam, and J. Kurose. Congestion-Aware Caching and Search in Information-Centric Networks. In *ACM SIGCOMM ICN*, 2014.

[4] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[5] S. Borst, V. Gupta, and A. Walid. Distributed Caching Algorithms for Content Distribution Networks. In *IEEE INFOCOM*, 2010.

[6] G. Cofano, L. De Cicco, and S. Mascolo. A Control Architecture for Massive Adaptive Video Streaming Delivery. In *ACM VideoNext Workshop*, 2014.

[7] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH). In *IEEE Packet Video Workshop (PV)*, 2013.

[8] M. Dehghan et al. On the Complexity of Optimal Routing and Content Caching in Heterogeneous Networks. In *IEEE INFOCOM*, 2015.

[9] A. Finamore. YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience. In *ACM SIGCOMM IMC*, 2011.

[10] Y. Jin, Y. Wen, and C. Westphal. Towards Joint Resource Allocation and Routing to Optimize Video Distribution over Future Internet. *IFIP Networking*, 2015.

[11] D. Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[12] D. H. Lee, C. Dovrolis, and A. C. Begen. Caching in HTTP Adaptive Streaming: Friend or Foe? In *ACM NOSSDAV Workshop*, 2014.

[13] C. Liu et al. Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. *Elsevier Signal Processing: Image Communication*, 27(4):288–311, Apr. 2012.

[14] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas. Video Delivery over Heterogeneous Cellular Networks: Optimizing Cost and Performance. *IEEE INFOCOM 2014*, 2014.

[15] Y. Sun, S. K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, and S. Uhlig. Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand Workloads. In *CoNEXT*, 2014.

[16] H. Susanto, B. Kim, and B. Liu. User Experience Driven Multi-Layered Video Based Applications. In *IEEE ICCCN*, 2015.

[17] L. Toni et al. Optimal Set of Video Representations in Adaptive Streaming. In *ACM MMSys*, 2014.

[18] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie. Optimal Cache Allocation for Content-Centric Networking. In *IEEE ICNP*, 2013.

[19] S. Yun, D. Kim, X. Lu, and L. Qiu. Optimized Layered Integrated Video Encoding. In *IEEE INFOCOM*, 2015.

[20] W. Zhang, Y. Wen, Z. Chen, and A. Khisti. QoE-Driven Cache Management for HTTP Adaptive Bit Rate Streaming Over Wireless Networks. *IEEE Trans. Multimedia*, 15(6):1431–1445, Oct. 2013.

MC3: A Cloud Caching Strategy for Next Generation Virtual Content Distribution Networks

Pietro Marchetta*, Jaime Llorca†, Antonia M. Tulino*†, Antonio Pescapé*‡
*Universitá di Napoli Federico II and ‡NM2 Srl, Italy. Email: {pietro.marchetta, pescape}@unina.it
†Bell Labs, Nokia, NJ, USA. Email: {jaime.llorca, a.tulino}@nokia.com

***Abstract*—With the advent of network functions virtualization and software defined networking, cloud content distribution network (CDN) providers can auto-scale their virtual CDN appliances in order to meet changing demands for commercial and user generated content services in a cost and energy efficient manner. However, existing caching policies, constrained to work with dedicated CDN resources and designed to maximize local cache hit rates, do not exploit the elasticity of virtualized cloud environments to adaptively guarantee service requirements with minimum cost. In this paper, we design and evaluate MC^3 (MinCostCloudCache), an adaptive distributed caching strategy whose fundamental goal is to guarantee content service requirements while minimizing the use and associated cost of the shared physical infrastructure. MC^3 estimates the global benefit of caching an object at a network node using only locally available information. The caching benefit is flexible and adaptive to the particular content service requirements, and is aware of the behavior of neighbor network caches, creating effective cache cooperation using only local information. Through simulation, we show how MC^3 not only reduces the experienced average delay with respect to existing caching policies, but it also uses significantly less storage and transport resources, leading to increased revenues and reduced operational costs.**

I. Introduction

Content distribution networks (CDNs) are highly distributed systems consisting of globally dispersed cache servers that allow hosting and delivering content items close to the end users, thus providing improved user experience and reduced transport costs. Traditional CDNs are composed of dedicated hardware appliances that the operator dimensions according to estimated peak demands. For a given dedicated CDN deployment, the extent to which the benefits of network caching can be obtained depends crucially on the efficacy of the implemented content caching strategy. Given that today's CDNs are composed of a relatively small number of fixed-size hierarchical caches, content caching strategies are typically designed to maximize local hit rates, or the fraction of requests served by a given cache, which has been shown to be a good proxy for average delay in hierarchical CDNs [1]. However, the increasing dynamics and heterogeneity of content types, popularity, and service requirements, challenge the efficiency of traditional CDNs, in which dedicated storage and delivery appliances need to be pre-provisioned based on estimated peak demands, resulting in excessive over-provisioning and/or degraded quality of service (QoS).

ISBN 978-3-901882-83-8

With current advances in network virtualization and programmability, network operators have the opportunity to design their content distribution solutions in the form of elastic virtual networks over a common cloud network infrastructure [2]-[4]. In this way, operators can adaptively optimize the combined use of storage and transport resources to meet application requirements with minimum cost. In this attractive scenario, existing caching solutions, designed to work with fixed-size dedicated CDN appliances, cannot exploit the flexibility of evolved virtualized cloud environments nor the properties of the different content services to adaptively guarantee service requirements with minimum use of the shared physical infrastructure. We therefore argue for a shift in the design of content caching strategies for future cloud CDNs, driven by the main goal of minimizing the overall network's operational cost while guaranteeing QoS requirements.

A. Contributions

In this paper, we look at content caching in the context of next generation virtual CDNs that can host a variety of content services and elastically scale their network resources. Our approach – instead of just maximizing local hit rates for typically small hierarchical fixed-size CDNs – aims at guaranteeing content services' QoS requirements with minimum overall use of the shared cloud network's infrastructure.

The contributions of this work are fourthfold. First, we analytically characterize the optimal cloud caching policy for a given first-order stationary input process (Sec. II and Sec. III). Second, based on the structure of the optimal stationary solution, we propose MC^3 (Minimum Cost Cloud Cache), a fully distributed cloud caching algorithm targeting optimal caching decisions based on local estimates of the global cost benefit (Sec. IV). MC^3 provides effective cache cooperation with negligible overhead via the adaptive learning of transport costs to access neighboring replicas and local content popularity. Specifically, with MC^3, network nodes: *i*) infer neighbors' actions from information in object arrivals; *ii*) infer local content popularity from information in request arrivals; and *iii*) compute the benefit of caching an object at a particular location based on the overall cost reduction it can provide. Third, we implement the proposed caching strategy in a custom-built discrete event simulator (Sec. V-A). Fourth, we perform a comparison with a number of well known caching strategies (LRU-LCE, Perfect-LFU, an Oracle), demonstrating the significant performance and efficiency gains that MC^3 can

provide in virtual CDN environments (Sec. V-B). We show the superiority of MC^3 in terms of average delivery delay and overall operational cost with varying transport-to-storage cost ratio, cache size, and content popularity settings.

B. Related Work

A substantial amount of research has been devoted to the content distribution problem (CDP), where the goal, in its most general setting, is to find the placement and routing of content objects in an arbitrary capacitated network that minimizes the combined transport and storage cost while satisfying possible delivery deadline constraints. The authors in [5] provide a comprehensive complexity classification of the CDP. Interestingly, while NP-Hard in general, the CDP is shown to be polynomial-time solvable in tree networks and in arbitrary networks that allow coding between objects of the same requested content. The work in [6] addresses the CDP in realistic AS-level topologies showing how the footprint of dedicated CDNs must expand to accommodate increasingly tighter user requirements. A number of works have addressed the design of approximated solutions for the CDP, mostly relying on LP-relaxation techniques (e.g., [7], [8], [9]) or greedy algorithms (e.g., [10], [11], [12]) that exploit special assumptions such as uniform object sizes, network symmetry, and hierarchical topologies. The solutions to the CDP are centralized and proactive, in the sense that content placement decisions are made based on global estimates of the users' average demands (e.g., content popularity) over a given time period, typically in the range of hours or even days. Network caches are then updated to best satisfy future requests over the given time period. The performance of centralized proactive solutions heavily depends on the system dynamics and its corresponding prediction accuracy. In fact, with the increasing volatility and unpredictability of next generation content services, errors in the popularity estimates and the overhead associated to frequent cache updates can significantly degrade the performance of centralized proactive solutions.

In highly dynamic and unpredictable scenarios, content distribution solutions must resort to distributed reactive algorithms that adapt to fast changes in content popularity with minimal overhead. An extensive line of work has also been devoted to the distributed dynamic content replacement problem, where the objective is to adaptively refresh the network caches as content objects travel through the network (e.g., [13], [14], [15]). The most common cache replacement algorithms are LRU (Least Recently Used) and LFU (Least Frequently Used), by which the least frequently/recently used content object is evicted upon arrival of a new object to a network cache. Due to its simplicity, LRU is the most widely used caching algorithm in today's CDNs and the most analyzed in the context of emerging paradigms such as ICN (see [16], [17] and references therein). These studies show the benefits of LRU-based caching to reduce dissemination latency and network load, and the little improvements provided by alternative caching policies proposed to date. However, as pointed out earlier, existing caching policies have been designed and compared against hit-rate performance metrics, ignoring the operational cost associated to the use of storage and transport resources. Hence, motivated by the increasing adaptability and programmability associated to the configuration of next generation cloud networks, and the dynamics and heterogeneity of next generation content services, we argue that centralized proactive content distribution solutions must be complemented with distributed reactive caching algorithms that are designed to achieve global system objectives, such as overall cloud network operational efficiency, via simple local interactions that incur minimal overhead.

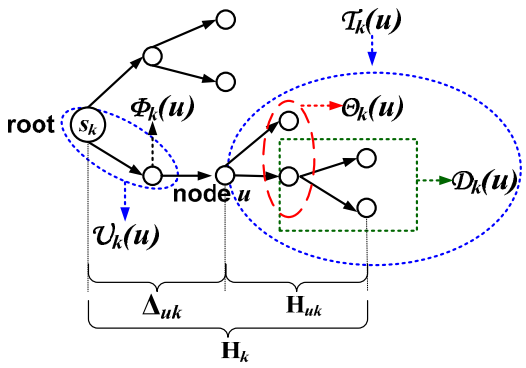


Fig. 1: The routing tree for object k, $\mathcal{T}_k \subset \mathcal{G}$, rooted at the source or repository of k, s_k, for a given time period. $\Phi_k(u)$ denotes node u's upstream neighbor in $\mathcal{T}_k$.

II. Network Model

In a distributed cloud network architecture, a virtual CDN consists of a set of virtual caches (vCache), implemented as virtual network functions (VNFs) in an NFV framework. The vCache nodes are interconnected by virtual links (vLink), representing the logical connectivity. We model a virtual CDN as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with V vCaches and E vLinks. We assume content items are partitioned into equal-size objects $k \in \mathcal{K}$ and denote by λ_{uk} the exogenous average request rate for object $k \in \mathcal{K}$ at node $u \in \mathcal{V}$ during a specified time period. We denote by e_u^{st} the per-object storage cost of vCache $u \in \mathcal{V}$ and e_{uv}^{tr} the per-object transport cost of vLink $(u, v) \in \mathcal{E}$.

Motivated by the different time-scales at which routing and caching operate, here we do not address routing optimization and assume that the goal is to design a caching strategy for a given routing policy. We denote by $\mathcal{T}_k = (\mathcal{V}, \mathcal{E}_k)$ the routing tree rooted at the source or repository of k, s_k, as shown in Figure 1. It will also be useful to define $\mathcal{T}_k(u)$ as the set of nodes in the subtree of $\mathcal{T}_k$ rooted at u, $\mathcal{D}_k(u)$ as the nodes downstream of u, $\mathcal{U}_k(u)$ as the nodes upstream of u, $\Theta_k^j(u)$ as the j-th hop downstream neighbors of u, $\phi_k^j(u)$ as the j-th hop upstream neighbor of u (since most of the time we will refer to the one hop neighbors of u, we denote $\Theta_k(u) \equiv \Theta_k^1(u)$ and $\phi_k(u) \equiv \phi_k^1(u)$), H_k as the height of $\mathcal{T}_k$, H_{uk} as the height of $\mathcal{T}_k(u)$, and Δ_{uk} as the depth of u in $\mathcal{T}_k$, as shown in Fig. 1. We refer to e_{uk} as the unit transport cost of link $(\phi_k(u), u)$.

III. Optimal stationary policy

In this section, we analytically characterize the optimal cloud caching policy under the setting of stationary request process and sufficiently large storage capacity. Note that the "sufficiently large storage capacity" is a reasonable assumption in a cloud CDN, for which virtual storage may be largely available, but whose usage comes at a cost. While these assumptions may not always hold in practice, the structure of the resulting optimal policy will show extremely useful in driving the design of the proposed general caching strategy described in Sec. IV.

Under the assumption of a first-order stationary request process, we focus on designing a stationary caching policy that minimizes the average CDN cost. Letting $\mathbf{x} = \{x_{uk}\}$ denote a stationary caching configuration, with $x_{uk} = 1$ if object k is cached at node u, and $x_{uk} = 0$ otherwise, we seek the caching configuration $\mathbf{x}^*$ that satisfies

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathcal{C}(\mathbf{x}), \tag{1}$$

where

$$\mathcal{C}(\mathbf{x}) = \sum_{u\in\mathcal{V}}\sum_{k\in\mathcal{K}} \left(e^{st}_u x_{uk} + e^{tr}_{uk}\lambda_{uk}\left(1 - x_{uk}\right)\right), \tag{2}$$

$$e^{tr}_{uk} = \sum_{j=0}^{h_{uk}-1} e^{tr}_{\phi^j_k(u)k}, \qquad (\phi^0_k(u) \equiv u), \tag{3}$$

$$h_{uk} = \sum_{j=1}^{\Delta_{uk}} j\, x_{\phi^j_k(u)k} \prod_{p=0}^{j-1}\left(1 - x_{\phi^p_k(u)k}\right). \tag{4}$$

The average CDN cost, $\mathcal{C}(\mathbf{x})$, in (2), is computed as the sum over all nodes and objects of two mutually exclusive terms. The first term is the storage cost if k is cached at u, and the second term is the transport cost incurred in fetching k from the closest upstream copy at a rate λ_{uk}, if k is not cached at u. The variable e^{tr}_{uk} denotes the cost of the path from u to the closest upstream node caching k, and h_{uk} is the number of hops from u to the closest upstream node caching k.

Note that without capacity constraints, i.e., under the assumption of "sufficiently large storage capacity", (1) can be solved independently for each object $k \in \mathcal{K}$. Let $\mathcal{C}_{uk}$ denote the total cost for the delivery of object k over the subtree $\mathcal{T}_k(u)$ when u stores a copy of k:

$$\mathcal{C}_{uk} = e^{st}_u + \mathcal{C}^{(0)}_{uk}, \tag{5}$$

where, for all $u \in \mathcal{V}$ and $h = \{0, \ldots, H_k\}$,

$$\mathcal{C}^{(h)}_{uk} = \sum_{v\in\Theta_k(u)} \mathcal{C}_{vk}x_{vk} + \left(e^{tr}_{vk}\lambda^{(h+1)}_{vk} + \mathcal{C}^{(h+1)}_{vk}\right)(1 - x_{vk}) \tag{6}$$

is the total cost of the subtree $\mathcal{T}_k(u)$ when the closest upstream node caching k is h hops from u, and

$$\lambda^{(h)}_{vk} = \lambda_{vk} + \sum_{w\in\Theta_k(v)} \lambda^{(h+1)}_{wk}\left(1 - x_{wk}\right) \tag{7}$$

is the aggregate rate of requests for k at node v when the closest upstream node caching k is h hops from v.

TABLE I: Summary of the main variables used for the analysis of OSC, in addition to the routing tree variables in Fig. 1.

$\{x_{uk}\}$	stationary cache configuration, with $x_{uk} = 1$ if object k is cached at node u, and $x_{uk} = 0$ otherwise
e^{st}_u	per-object storage cost of node u
e^{tr}_{uv}	per-object transport cost of link (u, v)
λ_{uk}	request rate of object k at node u
h_{uk}	number of hops from node u to the closest upstream node caching object k
e^{tr}_{uk}	transport cost of transferring object k to node u form the closest upstream node caching object k
$\lambda^{(h)}_{uk}$	rate of requests for object k at node u when the closest upstream node caching object k is h hops from node u
$\mathcal{C}_{uk}$	total cost for the delivery of object k over subtree $\mathcal{T}_k(u)$ when node u is caching object k
$\mathcal{C}^{(h)}_{uk}$	total cost for the delivery of object k over subtree $\mathcal{T}_k(u)$ when the closest upstream node caching object k is h hops from node u

In the following, we present OSC (Optimal Stationary Cache) - see Tab. I for the details on the adopted notation - a dynamic programming algorithm that computes the minimum cost for the delivery of object k over $\mathcal{T}_k$, $\mathfrak{C}_k$, as

$$\mathfrak{C}_k = \mathfrak{C}_{uk}\Big|_{u=s_k}, \tag{8}$$

with

$$\mathfrak{C}_{uk} = e^{st}_u + \mathfrak{C}^{(0)}_{uk}, \tag{9}$$

where, for all $u \in \mathcal{V}$ and $h = \{0, \ldots, H_k\}$,

$$\mathfrak{C}^{(h)}_{uk} = \sum_{v\in\Theta_k(u)} \min\left\{\mathfrak{C}_{vk},\ e^{tr}_{vk}\mu^{(h+1)}_{vk} + \mathfrak{C}^{(h+1)}_{vk}\right\}, \tag{10}$$

$$\mu^{(h)}_{vk} = \lambda_{vk} + \sum_{w\in\Theta_k(v)} \mu^{(h+1)}_{wk}\left(1 - \mathfrak{X}^{(h+1)}_{wk}\right), \tag{11}$$

and

$$\mathfrak{X}^{(h)}_{wk} = \begin{cases} 1 & \text{if } \ e^{tr,h}_{wk}\,\mu^{(h)}_{wk} + \mathfrak{C}^{(h)}_{wk} \geq \mathfrak{C}_{wk} \\ 0 & \text{otherwise} \end{cases}, \tag{12}$$

with $e^{tr,h}_{wk}$ defined as

$$e^{tr,h}_{uk} = \sum_{j=0}^{h-1} e^{tr}_{\phi^j_k(u)k}. \tag{13}$$

Note that (5)–(7) are the equivalent of (9)–(11) when evaluated at the solution of OSC given by (12).

The optimality of OSC is based on the following theorem.

Theorem 1: $\mathfrak{C}_k$, as defined in (8), is the minimum cost for the delivery of object k over $\mathcal{T}_k$ with average request rates $\lambda_{uk}, \forall u \in \mathcal{V}$.

Proof. In order to prove Theorem 1, we first state and prove the following Lemma.

Lemma 1: For all $u \in \mathcal{V}$, $\mathfrak{C}^{(h)}_{uk}$ is the minimum cost over $\mathcal{T}_k(u)$ for the delivery of object k, given that the closest upstream node caching object k is h hops away from u, *i.e.,* for all $h = \{0, \ldots, H_k\}$

$$\mathfrak{C}^{(h)}_{uk} = \min_{\mathbf{x}}\left\{\mathcal{C}^{(h)}_{uk}\right\}. \tag{14}$$

Proof. To prove Lemma 1, we proceed by induction on the tree height. Let $\mathcal{H}(j)$ be the set of nodes at height j in $\mathcal{T}_k$.

First, we prove that the claim of Lemma 1 holds at the bottom of the tree. In fact, for all subtrees rooted at $u \in \mathcal{H}(1)$,

$$\begin{aligned}
&\min_{\mathbf{x}} \left\{\mathcal{C}_{uk}^{(h)}\right\} = \\
&= e_u^{st} + \min_{\mathbf{x}} \sum_{v\in\Theta_k(u)} \left\{\mathcal{C}_{vk}x_{vk} + \left(e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\right)(1-x_{vk})\right\} \\
&= e_u^{st} + \min_{\mathbf{x}} \sum_{v\in\Theta_k(u)} \left\{e_v^{st}x_{vk} + \left(e_{vk}^{tr}\lambda_{vk}\right)(1-x_{vk})\right\} \qquad (15)\\
&= e_u^{st} + \sum_{v\in\Theta_k(u)} \min\left\{e_v^{st},\, e_{vk}^{tr}\lambda_{vk}\right\} \qquad (16)\\
&= \mathfrak{C}_{uk}. \qquad (17)
\end{aligned}$$

where (15) follows from $\mathcal{C}_{vk} = e_v^{st}$, $\mathcal{C}_{vk}^{(h)} = 0$ and $\lambda_{vk}^{(h)} = \lambda_{vk}$ for all leaf nodes $v \in \Theta_k(u) \subset \mathcal{H}(0)$ and for all $h = \{0,\ldots,H_k\}$; (16) follows from the fact that each term in the summation only depends on x_{vk}; and finally (17) follows from $\mathfrak{C}_{vk}^{(h)} = 0$, $\mathfrak{C}_{vk} = e_v^{st}$, and $\mu_{vk}^{(h)} = \lambda_{vk}$ for all leaf nodes $v \in \mathcal{H}(0)$.

Next, we prove that if for all $v \in \mathcal{H}(\ell)$, $\ell = 0,\ldots,(j-1)$,

$$\mathfrak{C}_{vk}^{(h)} = \min_{\mathbf{x}} \left\{\mathcal{C}_{vk}^{(h)}\right\}, \qquad (18)$$

then it also holds that

$$\mathfrak{C}_{uk}^{(h)} = \min_{\mathbf{x}} \left\{\mathcal{C}_{uk}^{(h)}\right\}, \quad \forall u \in \mathcal{H}(j). \qquad (19)$$

To this end, notice that using (6),

$$\min_{\mathbf{x}}\left\{\mathcal{C}_{uk}^{(h)}\right\} = \sum_{v\in\Theta_k(u)} \min\left\{\min_{\mathbf{x}}\{\mathcal{C}_{vk}\}, \min_{\mathbf{x}}\left\{e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\right\}\right\} \qquad (20)$$

$$= \sum_{v\in\Theta_k(u)} \min\left\{\mathfrak{C}_{vk}, \min_{\mathbf{x}}\left\{e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\right\}\right\} \qquad (21)$$

where (20) is due to the fact that when the closest upstream node caching k is h hops away from v, then the optimal configuration of $\mathcal{T}_k(v)$ can be found by solving independently the optimal configuration for each of the subtrees rooted at $w \in \Theta_k(v)$; and (21) follows from the induction step in (18).

Now, we prove by *reductio ad absurdum* that

$$\min_{\mathbf{x}} \left\{e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\right\} = e_{vk}^{tr}\mu_{vk}^{(h+1)} + \mathfrak{C}_{vk}^{(h+1)}. \qquad (22)$$

To this end, first note that the non-strict inequality always holds:

$$\min_{\mathbf{x}} \left\{e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\right\} \leq e_{vk}^{tr}\mu_{vk}^{(h+1)} + \mathfrak{C}_{vk}^{(h+1)}; \qquad (23)$$

in fact, the right hand side of (23) is the cost of the subtree $\mathcal{T}_k(v)$ when k is cached $h+1$ hops away from v plus the cost of the upstream link (u,v), computed for the caching configuration given by (12), while the left hand side is the minimum over all possible caching configurations of the above function. Next, let us verify that strict inequality in (23) leads to a contradiction. To this end, assume (23) is strict, and let $\bar{\mathbf{x}} = \arg\min_{\mathbf{x}} \left\{e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\right\}$. Since, $\min_{\mathbf{x}} \left\{\mathcal{C}_{vk}^{(h+1)}\right\} = \mathfrak{C}_{vk}^{(h+1)}$ by induction, and $\mathfrak{C}_{vk}^{(h+1)} = \mathcal{C}_{vk}^{(h+1)}\Big|_{\mathfrak{X}}$ from (8)-(12), then

$$\mathcal{C}_{vk}^{(h+1)}\Big|_{\bar{\mathbf{x}}} \geq \mathfrak{C}_{vk}^{(h+1)} = \mathcal{C}_{vk}^{(h+1)}\Big|_{\mathfrak{X}}. \qquad (24)$$

From (6) and (7), we now have that

$$\begin{aligned}
&e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\Big|_{\bar{\mathbf{x}}} = \\
&= e_{vk}^{tr}\lambda_{vk} + \sum_{w\in\Theta_k(v)} \min\left\{\mathcal{C}_{wk}\Big|_{\bar{\mathbf{x}}}, \left(\left(e_{vk}^{tr} + e_{wk}^{tr}\right)\lambda_{wk}^{(h+2)} + \mathcal{C}_{wk}^{(h+2)}\right)\Big|_{\bar{\mathbf{x}}}\right\} \\
&= e_{vk}^{tr}\lambda_{vk} + \sum_{w\in\Theta_k^{\mathcal{C}_1}(v)} \mathcal{C}_{wk}\Big|_{\bar{\mathbf{x}}} + \sum_{w\in\Theta_k^{\mathcal{C}_2}(v)} \mathcal{C}_{wk}\Big|_{\bar{\mathbf{x}}} \\
&\quad + \sum_{w\in\Theta_k^{n\mathcal{C}_1}(v)} \left((e_{vk}^{tr} + e_{wk}^{tr})\lambda_{wk}^{(h+2)} + \mathcal{C}_{wk}^{(h+2)}\right)\Big|_{\bar{\mathbf{x}}} \\
&\quad + \sum_{w\in\Theta_k^{n\mathcal{C}_2}(v)} \left((e_{vk}^{tr} + e_{wk}^{tr})\lambda_{wk}^{(h+2)} + \mathcal{C}_{wk}^{(h+2)}\right)\Big|_{\bar{\mathbf{x}}} \qquad (25)
\end{aligned}$$

where $\{\Theta_k^{\mathcal{C}_1}(v), \Theta_k^{\mathcal{C}_2}(v), \Theta_k^{n\mathcal{C}_1}(v), \Theta_k^{n\mathcal{C}_2}(v)\}$ is a partition of $\Theta_k(v)$ such that:

$$\begin{aligned}
\Theta_k^{\mathcal{C}_1}(v) &\equiv \{w \in \Theta_k(v) :\ \mathfrak{X}_{wk}^{st,h+2} = 0 \text{ and } \bar{x}_{wk}^{st} = 1\},\\
\Theta_k^{\mathcal{C}_2}(v) &\equiv \{w \in \Theta_k(v) :\ \mathfrak{X}_{wk}^{st,h+2} = 1 \text{ and } \bar{x}_{wk}^{st} = 1\},\\
\Theta_k^{n\mathcal{C}_1}(v) &\equiv \{w \in \Theta_k(v) :\ \mathfrak{X}_{wk}^{st,h+2} = 1 \text{ and } \bar{x}_{wk} = 0\},\\
\Theta_k^{n\mathcal{C}_2}(v) &\equiv \{w \in \Theta_k(v) :\ \mathfrak{X}_{wk}^{st,h+2} = 0 \text{ and } \bar{x}_{wk} = 0\}.
\end{aligned}$$

Based on how the four regions are defined, using (24), the induction step (18), and the definitions of $\mathfrak{C}_{vk}^{(h+1)}$, $\mathfrak{X}_{wk}^{st,h+2}$, $\lambda_{vk}^{(h+1)}$ and $\mu_{vk}^{(h+1)}$ as in (6)-(12), it can be shown that:

$$\begin{aligned}
e_{vk}^{tr}\lambda_{vk}^{(h+1)} + \mathcal{C}_{vk}^{(h+1)}\Big|_{\bar{\mathbf{x}}} &\geq e_{vk}^{tr}\lambda_{vk} + \sum_{w\in\Theta_k^{\mathcal{C}_2}(v)\cup\Theta_k^{n\mathcal{C}_1}(v)} \mathfrak{C}_{wk} \\
&\quad + \sum_{w\in\Theta_k^{\mathcal{C}_1}(v)\cup\Theta_k^{n\mathcal{C}_2}(v)} \left(e_{vk}^{tr} + e_{wk}^{tr}\right)\mu_{wk}^{(h+2)} + \mathfrak{C}_{wk}^{(h+2)}\Big) \\
&= e_{vk}^{tr}\mu_{vk}^{(h+1)} + \mathfrak{C}_{vk}^{(h+1)}. \qquad (26)
\end{aligned}$$

Using (26) and (23), we show that a strict inequality in (23) leads to a contradiction, and hence (22) is proved. Replacing (22) in (21) and using (10), (19) follows. ∎

Using Lemma 1, it immediately follows that:

$$\begin{aligned}
\min_{\mathbf{x}} \{\mathcal{C}_{uk}(\mathbf{x}_k)\} &= \left(e_u + \min_{\mathbf{x}} \left\{\mathcal{C}_{uk}^{(0)}\right\}\right)\Big|_{u=s_k} \\
&= e_u + \mathfrak{C}_{uk}^{(0)}\Big|_{u=s_k} \\
&= \mathfrak{C}_{uk}\Big|_{u=s_k} = \mathfrak{C}_k
\end{aligned}$$

which concludes the proof of Theorem 1. ∎

We note that the complexity of OSC is linear with the product of the number of nodes and the height of the tree, $O(VH_k)$, and thus can find the optimal configuration for the delivery of $\mathcal{K}$ over $\mathcal{G}$ in $\sum_{k\in\mathcal{K}} O(VH_k) \leq O(V^2\Delta)$, with Δ the diameter of $\mathcal{G}$. Furthermore, OSC admits a distributed implementation, which requires $O(H_k - H_{uk})$ information exchange between each node $u \in \mathcal{V}$ at height H_{uk} and its upstream node $\phi_k(u)$.

However, the optimality of OSC is constrained to the availability of sufficiently large storage capacity and the stationarity of the input request process. While, as stated earlier, large storage capacity may be available in cloud CDNs, the increasing dynamics of content service demands can degrade the performance of OSC in practice. In the following, we propose MC^3, a fully distributed dynamic caching policy that builds on the structure of OSC, to drive local caching decisions that adapt to the system dynamics, while completely eliminating the need for any explicit exchange of information between neighbor nodes. In particular, with MC^3, local caching decisions are based on the binary criterion described in (12), where information about the closest upstream content copies and the caching configuration of the downstream nodes is inferred from the dynamic arrivals of requests and objects themselves. A detailed description of MC^3 and its key mechanisms are provided in Sec. IV. Also, while omitted due to space limitations, it is worth mentioning that for a hierarchical topology with homogeneous resources at each layer, under a first-order stationary request process, it can be shown that MC^3 achieves the optimal steady-state configuration given by OSC.

IV. MC^3: Algorithm Design

In this section, we describe MC^3 (Min Cost Cloud Cache), an on-line fully distributed cloud caching algorithm that allows nodes to make local caching decisions based on real-time estimates of the global cost benefit. Recall that in a cloud CDN, the goal is to guarantee QoS requirements (e.g., average delivery delay) while minimizing the overall operational cost. Hence, in MC^3, objects only get cached if doing so contributes to the global system benefit by: *i)* reducing the combined transport-storage cost, or *ii)* reducing the average delay.

As illustrated by the structure of the optimal stationary policy, OSC, a caching decision for object k at node u at time t is essentially a trade-off between the cost incurred in writing and keeping object k in the cache of node u, and the cost incurred in fetching k from the closest upstream node that has already cached k. We remark that while the cost of writing and keeping an object at a network nodes is pure storage resource cost, the cost of fetching an object from the closest upstream copy captures both transport resource cost and QoS, since the further the closest copy is, the higher is the delay in delivering the object to the requesting user.

Based on this observation, we can evaluate at time t, the benefit of caching object k at node u, as the difference between the average transport cost needed to transfer object k to node u based on the current network conditions, and the storage cost involved in writing and keeping k at u, as:

$$\mathrm{CB}_{uk}(t) = e^{tr}_{uk}(t)\hat{f}_{uk}(t) - e^{st}_{u} \tag{27}$$

$$e^{tr}_{uk}(t) = \sum_{(u,v)\in\Gamma_{uk}(t)} e^{tr}_{uv} \tag{28}$$

In (27), (28), $\hat{f}_{uk}(t)$ represents an estimate of the aggregate rate of requests for object k at node u; $e^{tr}_{uk}(t)$ is the transport cost paid at time t to transfer object k to node u from its closest upstream copy along the path $\Gamma_{uk}(t)$; and e^{st}_{u} represents the cost needed to write and store an object over a time unit in the cache of node u.

Note that in the case of homogeneous resources, i.e., $e^{tr}_{uv} = e^{tr}, \forall (u,v) \in \mathcal{E}$, (27) reduces to

$$\mathrm{CB}_{uk}(t) = h_{uk}(t)\hat{f}_{uk}(t)e^{tr} - e^{st}_{u}, \tag{29}$$

where $h_{uk}(t)$ is the number of hops to the closest upstream node caching k at time t.

In general, in MC^3, e^{tr}_{uv} represents a generic cost of transporting an object over a link, which may include transport equipment CAPEX and OPEX, as well as link delay. We remark that in the case that the link delay model is load-dependent, $e^{tr}_{uv}(t)$ would be a function of t, indicating the dependence on the current load.

In MC^3, each vCache node maintains a data structure named *shadow cache*, where key metadata related to both cached and not-cached objects is stored in order to compute the global benefit of caching an object at any given time. Each entry in the shadow cache contains the following information:

1) Object Identifier
2) Estimated request inter-arrival time, $\widehat{\Delta t_{uk}}(t) = 1/\hat{f}_{uk}(t)$
3) Estimated transport cost to closest replica, $e^{tr}_{uk}(t)$

Note that the algorithm is based on two main estimates: *i*) the cost of fetching k from the closest copy at the time of the next request arrival, $e^{tr}_{uk}(t)$, and *ii*) the request inter-arrival time of object k, $\widehat{\Delta t_{uk}}(t)$. In order to locally estimate *i*), we propose to store an additional field inside the packets carrying the objects through the network: field E indicates the transport cost incurred by an object as it travels through the network since the last time it was cached. When an object arrives at a cache node, field E is increased to take into account the cost of transferring the object across the last traversed link. The obtained value is then used as $e^{tr}_{uk}(t)$ in Eq. (27) in order to compute the global benefit of caching the object. If the node decides to cache the object or the object has been already cached, field E is reset to 0. This approach allows nodes to share the information they need to compute (29) with negligible constant-size communication overhead. In order to locally estimate *ii*), every time node u receives a new request for object k, it updates the estimated request inter-arrival time in the shadow cache, $\widehat{\Delta t_{uk}}(t)$, based on a predictor. A simple approach is to adopt a moving average computed based on past request arrivals with a suitable window size, as used in LFU and its variants [15].

Algorithm 1 : MC^3

For every node $u \in \mathcal{V}$, $v = \Phi_k(u)$
if Request for object k at node u (time t) **then**
  if Object k in the cache **then**
    Forward k downstream (set $E = 0$)
    Update $\widehat{\Delta t}_{uk}(t)$ in shadow cache entry
    Compute $\mathrm{CB}_{uk}(t)$
    if $(\mathrm{CB}_{uk}(t) > 0)$ **then**
      Keep k in the cache and update its position based on $\mathrm{CB}_{uk}(t)$ (decreasing order)
    else
      Remove k from the cache
    end if
  else
    Update $\widehat{\Delta t}_{uk}(t)$
    Forward request upstream
  end if
end if
if Object k at node u from $v = \Phi_k(u)$ (time t) **then**
  Get E from packet
  Update $e^{tr}_{uk}(t) = E + e^{tr}_{vu}$
  Get $\widehat{\Delta t}_{uk}(t)$ from the shadow cache entry
  Recompute $\mathrm{CB}_{uk}(t)$
  if $(\mathrm{CB}_{uk}(t) > 0)$ **then**
    Cache k based on CB_{uk} (decreasing order)
    Set $E = 0$
    if Cache full **then**
      Remove last object (least cost benefit)
    end if
  else
    Set $E = e^{tr}_{uk}(t)$
  end if
  Forward k downstream (including E)
end if

By relying on the shadow cache and the information carried by the objects travelling through the network, each node is able to identify the subset of objects with the highest cost benefit. This result is achieved by maintaining a list of object entries sorted in terms of decreasing cost benefit. Objects are added and removed to this list every time their cost benefit is recomputed. Note that only objects with a positive cost benefit are potentially cached. This implies that depending on the ratio between transport and storage cost as well as the characteristics of the stream of object requests, nodes will make use of different portions of the available virtual cache space. Finally, we use a negative cost benefit for those objects for which we do not have information inside the shadow cache since we do not have enough information to compute the request inter-arrival time. The object is potentially cached only starting from the second received request.

In order to mitigate the impact of possible overestimates (too short) of request inter-arrival times, a timer is used to update the entries in the shadow cache if no request arrives within a guard time (set as a factor of the estimated inter-arrival time). This approach is useful to correct inaccurate or stale metadata such as the estimated next request arrival time. Note that underestimates of request inter-arrival times are naturally updated when the actual request arrives.

The pseudo-code in **Algorithm** 1 describes the procedures invoked by MC^3 upon *i*) a new request arrival, and *ii*) a new object arrival to a vCache node. Note that MC^3 exhibits constant-time computational complexity and constant-size communication overhead, as neither the number of computations nor the information objects carry grow with the number of nodes and objects in the system. Indeed, in MC^3, objects themselves carry how much cost they incur as they travel through the network. This allows vCache nodes to adaptively learn relevant system information, effectively creating cache cooperation with minimal overhead.

V. Experimental Analysis

We analyze the benefit of MC^3 in the context a 2-layer vCache hierarchy with Internet video workloads that exhibit different content types, daily viewing patterns, and object popularity. The main parameter settings are derived from the work in [18], as described in the following.

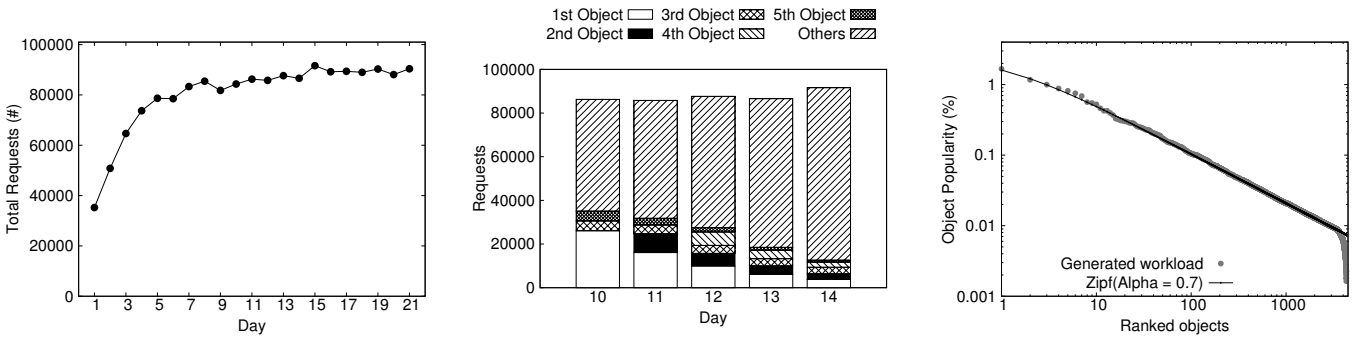
A. Simulation Methodology

Adopted topology. We consider a 2-layer tree structure of vCache nodes: three leaf vCache nodes are connected to a root vCache connected to the *library*, which stores all available content objects. User requests are first forwarded to the leaves in the hierarchy. A request is then forwarded to the root node or to the library only in case of a cache miss. The links between users and leaf vCache nodes are characterized by a delay of 20 ms, while all other links experience a delay of 50 ms. As in a number of previous works (e.g., [18], [19], [20]), we test the vCache hierachy against synthetic yet realistic streams of user requests for video objects.

Object types. We consider two types of video objects: *TV shows* and *Movies*. They differ in terms of size (Movies are typically twice as long as TV shows), and popularity trends (TV shows become unpopular much faster than Movies). The number of TV shows is typically much larger than the number of Movies: in our library, we adopt a shows-to-movies ratio of 4:1, as also suggested in [18].

Object requests. Video object requests are generated according to a Poisson process with average rate determined by the total number of requests to be generated during a specific portion of the day, as described in the following. In our experimentation, we generate $80,000$ requests every day, on average.

Daily pattern. The temporal evolution of the video object requests is known to show a clear time-of-the-day effect [19], with a peak during evening prime time and a lull during the night. To take into account this pattern, we partition the day into four time intervals: morning [6 a.m., 12 p.m.), afternoon [12 p.m., 6 p.m.), evening [6 p.m., 12 a.m.), and night [12 a.m., 6 a.m.). Letting R denote the total number of requests to generate during the day, we inject 10%, 20%, 30%, and 40% of the R requests during the night, morning, afternoon, and evening, respectively.



(a) Overall requests per day. (b) Requests per object. (c) Object popularity after 14 days.

Fig. 2: Details on the workload used in the simulation.

Object popularity. The stream of requests is generated as a series of independent trials drawn from a Zipf (or Zipf-like) distribution over the set of possible objects [15]. However, generating object requests according to a Zipf distribution does not capture the temporal evolution of the popularity of each object. Recently, Balachandran et al [19] observed a specific temporal trend in case of video objects. There is a peak of requests the first day the object becomes available, while the number of requests decreases exponentially during the following days. Hence, while the popularity of video objects over the entire observation period follows a Zipf distribution, a realistic workload must take into account this day-by-day temporal evolution of the object popularity. To achieve this effect, we follow the steps recently proposed by Akhtar et al. [18]: for each object k, *i*) we compute the total number of requests R_k over the entire observation period according to the object popularity extracted from the Zipf distribution; *ii*) given an observation period of N days, we randomly select the day X in which object k becomes publicly available: the R_k requests are then packed in the time range $[X,\ N]$; *iii*) to determine the first burst and the successive exponential decrease in the number of requests, we adopt a power series expansion such that the final number of requests generated over the time range $[X,\ N]$ is R_k. New objects are injected into the library at the beginning of the injection day causing the library to grow in size day by day. In our simulation, we inject 300 new video objects every day. Moreover, since empirical observations show that the popularity of TV shows decreases faster than that of Movies [19], we set the day-by-day popularity decrease rate to vary in $[0.3,\ 0.5]$ for TV shows and in $[0.05,\ 0.2]$ for Movies.

Fig. 2 shows an instance of the workload used in our simulations. We generate an average of $80,000$ requests per day, although we reach this value after a transitory period of 7 days (see Fig. 2a). Fig. 2b, provides a breakdown of the requests generated between the $10th$ and the $14th$ day. Considering the 5 most popular objects in this time range, we can notice that: *i*) the first, third, and fifth most popular objects are already available on the $10th$ day, whereas the second and forth objects become available only starting from the $11th$ and $12th$ day, respectively; *ii*) the number of requests for each of these objects decreases rapidly day after day. Despite this dynamic evolution in the number of requests, the object popularity computed over the first 14 days is very close to the expected Zipf distribution with Zipf parameter $alpha$ set to 0.7 (see Fig. 2c).

Observation period. We consider a long observation period of 21 days, *i.e.,* 3 weeks. At the end of this period, our library contains 6300 video objects.

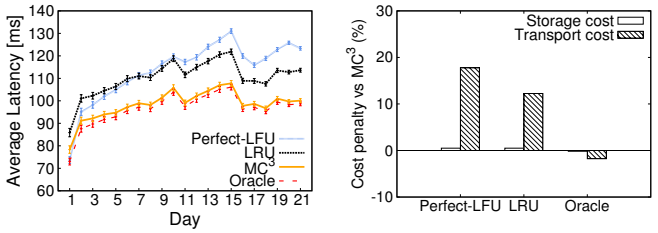
MC^3 settings. We adopt a moving average approach to estimate the object request inter-arrival and, after a first tuning phase, we set this weight to 0.5. In our experiments, we observe a negligible impact of slight modifications to this value on the overall vCDN performance.

Other caching strategies. We compare MC^3 with other caching strategies: *i)* LRU-LCE (*Least Recently Used - Leave Copy Everywhere*) – each cache node applies a least recently used replacement policy; *ii)* Perfect-LFU (*Perfect Least Frequently Used*) – each cache node applies a least frequently used replacement policy that tracks the number of requests for all objects in a shadow cache. This solution is known to well approximate the optimal hit-rate in case of static object popularity; *iii)* Oracle – each cache node can take omniscient caching decisions since nodes are informed about the future object popularity of each day. LRU and LFU represent simple and effective caching strategies made available in commercial products such as Apache Traffic Server, Squid, and Varnish, solutions widely adopted in operational environments.

We remark that the cache management cost of MC^3, while slightly higher than LRU – the lightest caching policy – is exactly the same as that of LFU or any other policy that is ordering-based (i.e., objects are kept in a specified order in the cache) and shadow-cache-based (i.e., policies that track non-cached objects' metadata).

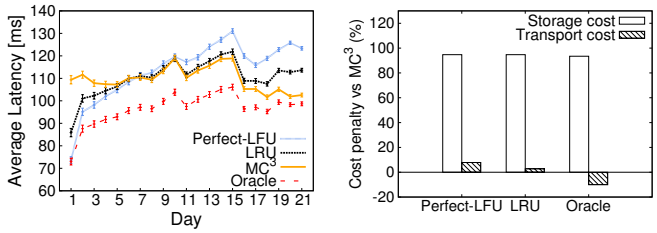
B. Experimental Results

We now describe the performance of MC^3 by varying the transport-to-storage cost ratio, cache size, and object pop-



(a) End to end latency. (b) Operational cost after 21 days.

Fig. 3: Performance when the transport-to-storage cost ratio is 10,000:1.



(a) End to end latency. (b) Operational cost after 21 days.

Fig. 4: Performance when the transport-to-storage cost ratio is 2,000:1.

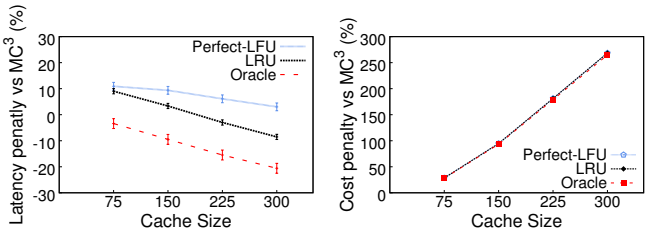
ularity. As done in similar works [22], [21], we relied on Omnet++ [23] to instrument our simulation.

Performance with a varying transport-to-storage cost ratio. We evaluate the caching strategies with a transport-to-storage cost ratio, e^{tr}_{uv}/e^{st}_{u}, of 10,000:1 and 2,000:1. Recall that e^{tr}_{uv} is used to capture not only transport resource costs but also QoS related penalties such as average delay. In this setting, we instrument each node in the hierarchy to cache no more than 50% of the objects injected every day (*i.e.,* 150 objects), while the popularity of each object is generated from a Zipf distribution with Zipf parameter $alpha$ equal to 0.7.

Fig. 3 shows the performance achieved when the ratio is 10,000:1. Fig. 3a shows the daily average latency, *i.e.,* the time the cache hierarchy takes to deliver a requested object to the user. For most of the caching strategies, the performance stabilizes after 7 days. This result is expected since the library is empty at the beginning of the simulation and gets filled day after day with new objects (see Fig. 2a): the equilibrium between previously injected unpopular objects and recently injected highly popular objects is reached only after this initial transitory. Hence, in the following, we discuss the average performance achieved between the $7th$ and $21th$ day. In this setting, the performance of MC^3 is very close to Oracle, the cache strategy that takes omniscient caching decisions. On an average day, the hierarchy instrumented with MC^3 delivers objects to the users with a latency only 1.8% higher than using Oracle, taking 12 ms and 12.4 ms less than LRU and Perfect-LFU, on average. This result is a direct consequence of the higher hit rate achieved by MC^3. Indeed, the average combined hit rate achieved by MC^3 is 28.3% and 33.3% higher compared to LRU and Perfect-LFU, respectively. At the same time, MC^3 guarantees a lower operational cost. Fig. 3b shows the total transport and storage cost penalty paid when using the other caching strategies relative to MC^3. Due to the high transport-to-storage cost ratio, MC^3 uses almost all the available space at the cache nodes in this setting. For this reason, we observe only a limited gain in terms of storage cost. On the other hand, MC^3 carefully selects which objects to cache according to the cost of transferring them over the network, thus achieving a significant gain in terms of total transport cost. Indeed, the transport cost registered when using LRU and Perfect-LFU is 17% and 12% higher than MC^3, respectively. Hence, for high transport-to-storage cost ratio, MC^3 achieves higher performance compared to LRU and Perfect-LFU in terms of hit rate and latency, with a similar storage cost, but a much lower transport cost.

Fig. 4 shows the performance achieved by the tested caching policies in the case of a transport-to-storage cost ratio of 2,000:1. Note that MC^3 is the only strategy modifying its behavior: in this setting, the latency achieved by MC^3 is similar to that achieved by LRU, and lower than that of Perfect-LFU. At the same time, MC^3 achieves this result by using only a fraction of the cache space available at each node: while all other strategies fully use the entire available storage space, MC^3 uses on average only 47% of the space in each cache node. The direct consequence is a significantly lower total operational cost as reported in Fig. 4b: using LRU (Perfect-LFU) leads to a total storage cost of 92% (93%) higher, and a total transport cost 8% (3%) higher than when using MC^3. In conclusion, for less unbalanced ratio between transport and storage cost, MC^3 is able to provide similar or higher performance than LRU and Perfect-LFU in terms of hit rate and latency, with a much lower storage and transport cost.

Performance with a varying cache size. Fig. 5 reports the results achieved when varying the cache size. Each node is configured to cache up to 25%, 50%, 75% and 100% of the amount of objects injected every day, *i.e.,* 75, 150, 225, and 300 objects, respectively. Fig. 5a shows the latency penalty paid when using all other caching strategies relative to MC^3: for larger cache sizes, the latency gain of MC^3 decreases. This happens because the other caching strategies fully use the available storage space in the cache disregarding the associated operational cost, while MC^3 keeps caching objects according to the transport-to-storage cost ratio (2,000:1 in this setting). Fig. 5b shows the cost penalty of operating the cache hierarchy with these strategies compared to using MC^3. The total operational cost increases sharply with the cache size: compared to all the other strategies, MC^3 guarantees savings that go from 28% up to 264% with the increasing cache size. Finally, by being aware of the transport and storage relative costs, MC^3 is able to select the objects to cache in order to guarantee a reasonable average latency while significantly saving in the overall operational cost. Note that one may easily



(a) Average end to end latency. (b) Total operational cost.

Fig. 5: Performance compared to MC^3 when the transport-to-storage cost ratio is 2,000:1 for different cache sizes.

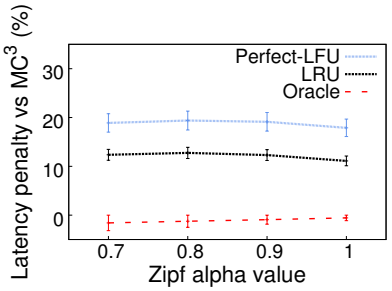


Fig. 6: Latency penalty compared to MC^3 when the transport-to-storage cost ratio is 10,000:1 for different object popularity.

improve the latency performance of MC^3 by simply increasing the transport-to-storage cost ratio to induce the MC^3 nodes to cache more objects, leading to higher hit rate, lower latency, but at the expense of higher operational cost.

Performance with varying popularity. Finally, we also investigate whether and how the performance of MC^3 changes when varying the object popularity. Results are reported in Fig. 6. We consider a cache size of 150 objects and a transport-to-storage cost ratio of 10,000:1. In this setting, the MC^3 nodes are induced to fully use their available storage space. We vary the Zipf parameter $alpha$ in the range $[0.7, 1.0]$. On average, we observe an almost constant gain over LRU and Perfect-LFU, with a latency reduction of 12% and 18%, respectively. Note that the latency achieved by Oracle is only very slightly lower than the one achieved by MC^3.

VI. Conclusions

Motivated by the dynamics and heterogeneity of next generation cloud-based CDNs, and the crushing burden that content storage and transport costs pose on cloud network operators, in this paper we took a fresh look at the dynamic content distribution problem from an overall cost-oriented perspective. We proposed a novel fully distributed online caching solution, we called MC^3, aiming at guaranteeing QoS requirements with minimum overall use of the shared cloud network's infrastructure. We first analytically characterized the optimal cloud caching policy for a given first-order stationary input process, and then – inspired by the structure of the optimal stationary solution – we developed MC^3, an online caching policy that guides local caching decisions based on real-time estimates of the global cost benefit. We implemented MC^3 in a custom-built CDN simulator and presented performance results for different settings of the transport-to-storage cost ratio, cache size, and object popularity. We also provided a comparison with three well known caching strategies (LRU-LCE, Perfect-LFU, and an Oracle), demonstrating the significant performance and efficiency gains – in terms of average latency and overall operational cost – that MC^3 can provide in virtual CDN environments.

References

[1] H. Che, Y. Tung, Z. Wang, "Hierarchical Web caching systems: modeling, design and experimental results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, 2002.

[2] Bell Labs Strategic White Paper, "The Programmable Cloud Network - A Primer on SDN and NFV," June 2013.

[3] Marcus Weldon, "The Future X Network," *CRC Press*, October 2015.

[4] J. Llorca, C. Sterle, A. M. Tulino, N. Choi, A. Sforza, A. E. Amideo, "Joint Content-Resource Allocation in Software Defined Virtual CDNs," *IEEE ICC'15 CCSNA Workshop*, London, England, 2015.

[5] J. Llorca, A.M. Tulino, "The content distribution problem and its complexity classification," *Bell Labs technical report*, 2013.

[6] S. Hasan, S. Gorinsky, C. Dovrolis, and R. Sitaraman, "Trade-offs in Optimizing the Cache Deployments of CDNs", *IEEE INFOCOM'14*, pp. 460-468, 2014.

[7] I.D. Baev, R. Rajaraman, C. Swamy, "Approximation algorithms for data placement in arbitrary networks," *ACM SODA'01*, 2001.

[8] I.D. Baev, R. Rajaraman, C. Swamy, "Approximation algorithms for data placement problems," *SIAM Journal on Computing*, vol. 38, pp. 1411-1429, 2008.

[9] S. Borst, V. Gupta, A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," *IEEE INFOCOM'10*, San Diego, 2010.

[10] P. Krishnan, D. Raz, Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. on Networking*, vol. 8, no. 5, pp. 568–582, 2000.

[11] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," *IEEE INFOCOM'01*, vol. 3, 2001.

[12] M.R. Korupolu and M. Dahlin, "Coordinated placement and replacement for large-scale distributed caches," *IEEE Transactions on Knowledge and Data Engineering*, vol.14, pp. 1317–1329, 2002.

[13] Wang, J., "A survey of web caching schemes for the internet," *ACM SIGCOMM CCR*, v. 29, n. 5, pp. 36-46, '99.

[14] P. Cao, S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," *Usenix symposium on internet technologies and systems*, vol. 12, no. 97, pp. 193–206, 1997.

[15] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," *IEEE INFOCOM'99*, vol. 1, pp. 126–134, 1999.

[16] G. Carofiglio, M. Gallo, L. Muscariello, D. Perino, "Modeling data transfer in content-centric networking," *IEEE ITC'11*, pp.111–118, 2011.

[17] E. J. Rosensweig, J. Kurose, "A Network Calculus for Cache Networks," *IEEE INFOCOM'13*, pp. 85–89, 2013.

[18] S. Akhtar, A. Beck, I. Rimac, "HiFi: A Hierarchical Filtering Algorithm for Caching of Online Video," *Proc. of the 23rd ACM international conference on Multimedia (MM '15)*. ACM, NY, USA, 421-430.

[19] A. Balachandran, V. Sekar, A. Akella, and S. Seshan. "Analyzing the potential benefits of CDN augmentation strategies for Internet video workloads." *ACM SIGCOMM IMC*, pp. 43-56. 2013.

[20] B. Paul and C. Mark, "Generating representative web workloads for network and server performance evaluation," *Proc. ACM SIGMETRICS*, Madison, USA, 1998.

[21] J. Llorca, A. M. Tulino, K. Guan, J. Esteban, M. Varvello, N. Choi, D. C. Kilper, "Dynamic in-network caching for energy efficient content delivery," *IEEE INFOCOM'13*, Turin, Italy, 2013.

[22] K. Stamos, G. Pallis, A. Vakali, D. Katsaros, A. Sidiropoulos, and Y. Manolopoulos. "CDNsim: A simulation tool for content distribution networks." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 20, no. 2 (2010): 10.

[23] A. Varga, "The OMNeT++ discrete event simulation system." *ESM 2001*, vol. 9, no. S 185, p. 65. sn, 2001.

Enhanced Caching Strategies At the Edge of LTE Mobile Networks

Andre S. Gomes *†, Torsten Braun *, Edmundo Monteiro †
*Institute of Computer Science, University of Bern, Switzerland
{gomes,braun}@inf.unibe.ch
†CISUC, University of Coimbra, Portugal
{asng,edmundo}@dei.uc.pt

***Abstract*—With a boom in the usage of mobile devices for traffic-heavy applications, mobile networks struggle to deliver good performance while saving resources to support more users and save on costs. In this paper, we propose enhanced strategies for the preemptive migration of content stored in Information-Centric Networking caches at the edge of LTE mobile networks. With such strategies, the concept of content following the users interested in it becomes a reality and content within caches is more optimized towards the requests of nearby users. Results show that the strategies are feasible, efficient and, when compared to default caching strategies, ensure that content is delivered faster to end users while using bandwidth and storage resources more efficiently at the core of the network.**

***Index Terms*—Information-Centric Networking, Content Migration, Caching, LTE.**

I. Introduction

Mobile network evolution in the last few years has been quite intense, with major increase of throughput performance and resources usage efficiency. Such evolution is mostly driven by tremendous demand of bandwidth [1], on the one hand because smartphones and other mobile devices play a major role as content demanders, and on the other hand because traffic-heavy applications are part of the daily life of millions of people. However, satisfying the content requirements of the current number of users with such dynamic networks is still an open challenge, which is currently being addressed by a number of emerging concepts and technologies.

As far as the network is concerned, new 5G concepts such as Network Function Virtualization (NFV) [2] are emerging, allowing mobile networks to adapt more dynamically to different conditions and requirements, and also to support other value-added technologies. One of these efforts is Cloud Radio Access Network (C-RAN) [3][4]. It brings the possibility to virtualize the entire 3GPP Long Term Evolution (LTE) radio infrastructure, except for the antennas. Virtualized infrastructures extend the cloud computing concept to the Radio Access Network (RAN), and explore the modularity of the components together with the usage of general-purpose hardware infrastructure to run evolved Node Bs (eNBs). Such fact transforms C-RAN into an enabler for deployment of value-added services closer to the edge of mobile networks, i.e. in very close proximity to mobile users. Despite increased delays due to its characteristics, the proximity deployment of other services allows for performance gains and cost savings that more than surpass those overheads and improve the end-to-end service.

ISBN 978-3-901882-83-8 © 2016 IFIP

In this direction, Future Internet (FI) concepts such as Information-Centric Networking (ICN) [5], which proposes a change in the current host-centric paradigm of requesting content, are becoming increasingly important due to the advantages brought together by its content-centric architecture. Namely: performance improvements [6], indirect bandwidth savings from its caching-based architecture, enhanced mobility support [7] and increased security [8].

With such concepts in mind, proposals appear to take advantage of the fact that they complement each other. Gomes et al. [9] evaluate the feasibility of deploying ICN together with 3GPP LTE mobile networks, leveraging the C-RAN concept and its role as an enabler for the deployment of additional services at the edge of these mobile networks. In that work, authors conclude that there are clear benefits of deploying ICN routers co-located with LTE eNBs, such as bandwidth savings at the core network and lower latency to retrieve content derived from the proximity to end users. Those findings are also in line with works such as [6], and show that there is an important demand of enhanced caching strategies to have content cached closer to the users interest in it while using resources efficiently.

Those caching strategies are twofold: first they are used to populate edge caches, and thereafter they must maintain content where it will yield the most benefit at any given time. As users are increasingly mobile and tend to move between different locations quite often, it is safe to assume that what is cached at a location is not necessarily what is going to be requested by the users that will be there in the next few hours or days. Studies [10] even show that user interests in social media content contribute deeply to its locality and homophily characteristics, which means that people geographically close to each other may have common or similar interests of content objects (locality) [11] and also that users are clustered by regions and interests (homophily) [12]. That leads to the question: how should caching strategies handle user mobility? Such question does not have a simple answer, as some assumptions have to be made and challenges need to be considered to reach a preliminary conclusion. First, it is important that user mobility is predicted to perform preemptive actions, and proposals

exist to deal with it [13][14][15]. Then, if a set of users is predicted to be at a location, more complex decisions need to be made in order to have accurate migrations that minimize overhead and maximize performance. The first decision is whether content from caches at the origin of the users should be migrated to other caches at the possible destinations. Once that is established, other questions arise: where should content be migrated to (mobility prediction usually outputs a list of possible destinations with different probabilities), which subset of the content should be migrated, how it should be migrated and when should it be migrated.

In this paper, we attempt to answer the previously described questions by developing content migration strategies that handle the required decisions and deliver the greatest possible trade-off between benefit and cost. In section II, existing proposals to address content migration strategies are analyzed. Section III introduces our proposal for the migration of content and related decisions. Section IV describes experimentation scenarios for the evaluation of the proposal. Section V presents the results of the performed experiments. Finally, in section VI, the main achievements of this work are highlighted.

II. Related Work

When considering strategies that take into account the mobility of users to decide on placement/migration of services or content within mobile networks, only a few works exist and there are still a number of shortcomings to be addressed.

One proposal assumes that mobility of the user is considered for services placement and scaling [16]. In this case, orchestration of distributed cloud services is done by predicting user mobility, i.e. more or less resources are allocated if the system predicts that users will move to/from the location of each small Data Center (DC). However, migration of services from one location to another is not considered.

In this direction, one very important concept towards migration strategies - Follow-Me Cloud - was first proposed by Taleb et al. [17]. It essentially considers that small DCs are present closer to the edge of mobile networks and proposes that services are deployed in close geographical proximity to users. Hence, when users move to a different location, those services should be migrated and follow the user. To handle the decision, several different models can be used. An analytical model based on Markov Decision Processes is proposed [18][19].

Such model considers that user positions must be found in order to have services instantiated in the optimal DC. It relies on the random walk mobility model to try to predict future positions, and when the user is n hops away from its current optimal DC, migrations are triggered using a system modeled with Continuous-Time Markov Chains. Also, when considering if data migration should be done, factors such as class of the user, load policies, service migration costs and service migration duration need to be analyzed. Bearing these factors in mind, it is assumed that cost and service disruption are to be minimized, and the user should be connected to the optimal DC as often as possible. This approach provides many benefits for the migration of services, but only a final destination is considered, not multiple destinations along a path. Moreover, it does not decide which services to migrate, it only considers a single user (overhead of migration for a single user may not be justified) and does not deal with specificities of migrating content or even stateful services.

As far as strategies to deal with content migration are concerned, other works [20] have looked at the problem in Peer-to-Peer (P2P) networks from the provider perspective. With a typical hierarchical Content Delivery Network (CDN) architecture, the main requirement is to distribute content among nodes in a way that leaf nodes get the most traffic and root nodes are seldom used. This strategy increases performance and thus reduces latency for end users, and relies on decisions to migrate/copy content from one node to another depending on popularity and cost. However, as it maximizes the usage of caches while attempting to maximize performance, those decisions are a NP-complete problem that is hard to manage. In very dynamic mobile networks, that poses an issue due to the need of quick and proactive decisions sometimes even before there is user movement, and that is why other works aim at less complex and local approaches that maintain hierarchical caches efficiently used [21].

Another proposal [22] takes dynamic mobile networks into account, and uses proactive migration strategies for content, i.e. migration is triggered when it is predicted that the user will move to a neighbor location. Using a proxy system, it is proposed that subscribed content is pre-fetched whenever it is predicted that a user will move to the geographical region of another proxy. With the knowledge of possible destinations and corresponding probabilities, a decision has to be made in order to select the destination proxies for the content while minimizing cost (migration cost and cache storage) and maximizing benefit (latency and cache hit ratios). Despite some gains in terms of delay, the number of criteria for migration decisions is small and no different weights are considered, there are no replacement policies when caches become full and the required single user mobility prediction is too simplistic/naive, i.e. the effect of a single user on the entire network is questionable when comparing to the required overhead of content migration and cache usage.

Considering all the proposals and the issues they fail to address, we propose a system that relies on their positive findings and at the same time attempts to address the challenges not taken into consideration.

III. Enhanced Caching Strategies

In-line with the idea of Follow-Me Cloud (FMC) described in the previous section, the proposal can be summarized into making decisions and perform preemptive migrations of mobile network's edge-cached content based on user mobility, i.e. migrations (copies) ahead of future user requests at a new location. The following key objectives are assumed: mobility prediction must be used to take actions before users move from one location to another, content may only be migrated if it is likely it will yield benefit at the destinations, multiple

destinations may be considered at the same time to improve accuracy and migration cost should be minimized as much as possible. We also assume that content migrations should happen among caches with modified policies, as they typically implement a Least Recently Used (LRU) policy that can delete recently added content in a matter of seconds if the load of received Interests is high. In this case we are interested in maintaining our own policy, i.e. popularity based queue, with the most popular and recently accessed content at the head of the queue and deletions happening at the tail.

A. Architecture

In order to achieve the goals associated with the objectives of this proposal, an architecture was defined as illustrated in Fig. 1.

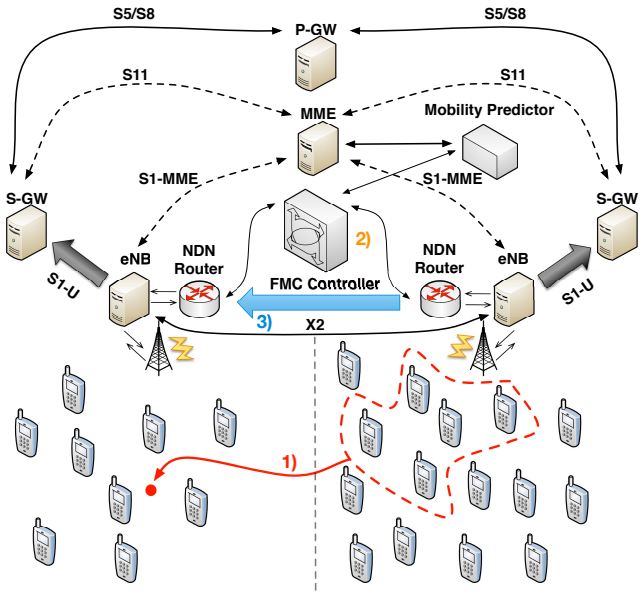


Fig. 1: Follow-Me Cloud Architecture

In this architecture, we assume as base architecture the 3GPP LTE Evolved Packet System (EPS). Namely, its main components (eNB, S-GW, P-GW, MME) and its main interfaces (X2, S1-U, S1-MME, S11, S5/S8). At the same time, a recent and increasingly popular approach for ICN is selected - Named Data Networking (NDN) [23]. We consider that NDN routers are co-located with 3GPP LTE eNBs [9], serving the subset of users present at each network cell. At the same time, information about the network and its users is gathered at the FMC Controller. This information includes mobility prediction data, local content popularity and availability at a given cell, number of users per cell and availability of resources such as storage. With that information, a multi-step process for content migration is triggered every time mobility is predicted or detected by the Mobility Predictor using input from the 3GPP LTE Evolved Packet Core's (EPC) Mobility Management Entity (MME), based on the defined time period between predictions:

1) FMC Controller is notified about user mobility (e.g. user ID n is moving from cell ID x to cell ID y with probability z) and decides, upon policies such as the number of users moving to a destination cell, if other steps should be taken or the process should stop.
2) If decisions have to be made regarding content migration, the FMC Controller has to decide: where to migrate content (cell ID and corresponding NDN router), what subset of the local content should be migrated (content object's prefixes), when to do it (according to other scheduled migrations) and finally how to do it (routing and load balancing for content requests).
3) After a decision is made, the FMC Controller issues a NDN message called Request of Interests that instructs the destinations' NDN router(s) to fetch the subset of content to be migrated from the closest source and place it at its cache.

B. Decision Techniques

As far as decisions are concerned, two types of decisions must be carefully analyzed and made in the proposed system. The first decision is **where** to migrate content when users are moving. Although mobility prediction will output a list of candidate destinations and respective probabilities, such set of destinations may not be complete and still needs to be reduced and ranked according to other important factors.

To increase the list of candidates, one may assume that neighbor cells in-between the returned destination candidates and the origin should also be considered. After all, the user will need to travel through those cells and, depending on the delta time, i.e. amount of time in the future considered for prediction, it may even stop and stay there longer than at the final destination. With this full list of destination candidates, the problem is now how to rank these in order to select only a few that satisfy the defined criteria and will yield the highest trade-off between benefit and cost.

Multiple-Criteria Decision-Making (MCDM) [24] is an approach to make decisions in the presence of multiple, usually conflicting, criteria. As it is not tied to a specific problem, it can be applied to a very diverse range of scenarios and problems, from business decisions to complex science problems. At the same time, it supports multiple weighted criteria and typically returns a finite number of solutions when dealing with a selection/assessment problem. Therefore, it fits the decision to be performed in terms of ranking/selection the destinations for content migration.

To handle decisions, which involve ranking of candidates, the most common methods are score methods. Within these, perhaps the most well-known and used method is Analytical Hierarchy Process (AHP) [25][26][27]. This method starts by summarizing the problem, deciding the hierarchical list of criteria to be considered for the decision and listing the alternatives to be ranked. In this case, the problem is already defined: a destination or destinations need to be selected for content migration. Considering that not only the destination of users is important but also to maximize the efficiency of cache usage within the network edge, the following criteria were defined:

- Mobility prediction information containing probabilities for destinations.
- Percentage of non-intersecting content between origin and alternative destination.
- Percentage of free storage space at alternative destination.
- Relative size of mobile group, defined as the ratio between number of users moving and users present at alternative destination.
- Cost of migration, estimated as the network transfer delay to copy the expected data size from its origin to an alternative destination.

Afterwards, a $N \times M$ matrix is created, where N is the number of alternatives and M the number of criteria. For each cell of the matrix, a score value is calculated to reflect how good the alternative is in terms of the criteria being considered.

As each of the criteria may have a different significance for the decision, each of them should also have a weight value to be considered. In order to rank criteria, judgment is used by creating a $M \times M$ matrix where each criteria is compared against the others using pair-wise comparisons, i.e. each compared to all the others in terms of importance. For instance, we may define that mobility prediction information is three times more important than the relative size of the mobility group. In that case, the cell that compares mobility prediction with relative size of the mobility group will have a value of 3/1, and the opposite comparison the value of 1/3. This matrix, however, cannot be used directly. An eigenvector with the final weights has to be calculated following the procedure:

1) Convert fractions to decimals.
2) Square the resulting matrix.
3) Sum up the rows of the matrix and get a vector. Each of the rows of the vector must be divided by the sum of all its rows to normalize the values.
4) Repeat the previous steps until the resulting vector is not different from the previously obtained vector.

With the scores of each alternative for each criterion and the weights, the score of alternative i is given by:

$$S_i = \sum_{\substack{j=1 \\ \forall i \in [1,N]}}^{M} w_j r_{ij} \tag{1}$$

where:

S_i is the score of the i^{th} alternative;

r_{ij} is the normalized rating of the i^{th} alternative for the j^{th} criterion, which is calculated as $r_{ij} = x_{ij}/(\max_i x_{ij})$ for benefits and $r_{ij} = \frac{1}{x_{ij}}/(\max_i \frac{1}{x_{ij}})$ for costs;

x_{ij} is an element of the decision matrix, which represents the original value of the j^{th} criterion of the i^{th} alternative;

w_j is the weight of the j^{th} criterion;

M is the number of criteria;

N is the number of alternatives.

With the list of destination alternatives ranked and sorted in descending order by their score, hereafter just called "ranking", the decision about where to migrate content may be made based on the defined policies. For instance, the first three alternatives (destinations) of the ranking can be selected and content will be migrated to all of them. Or, depending on the problem and assuming that the score is normalized, alternatives with scores above 0.75 are to be selected.

After the decision about the destinations has been taken, the remaining decision of **what** content should be migrated still needs to be taken. This decision has to be made considering that currently the association between LTE users and NDN users is not known, and therefore content cannot be related to a particular moving user. Therefore, it takes the local popularity of content as key criterion [21] and considers the following steps. First, if the content object being considered is available nearby (1 hop distance) or already at the destination, it will not be migrated. Second, if it is not available, and if free space is available at the destination, content objects are just migrated until the cache is filled. Third, if no free space is available, both popular content at the origin and destination should be considered together and ranked to fill the destination cache with the content that will deliver the greatest benefit for all the users (existing and new ones). That problem can be modeled as a Knapsack problem, and be solved with Dynamic Programming [28]. However, it is a NP-complete problem and, even if a solution is found, it may take too long to calculate. Therefore, another simpler approach was followed. The following equation was considered to calculate the score of each content object k:

$$S_k = \begin{cases} p_k * \dfrac{n_{mgt}}{n_{mgt} + n_{dst}}, \text{if the } k^{th} \text{content object is} \\ \text{considered for migration.} \\ \\ p_k * \dfrac{n_{dst}}{n_{mgt} + n_{dst}}, \text{otherwise.} \end{cases} \tag{2}$$

where:

S_k is the score of the k^{th} content object;

p_k is the local popularity of the k^{th} content object;

n_{mgt} is the number of users migrating from origin to the selected destination;

n_{dst} is the number of users at the selected destination for content.

With the content objects ranked and sorted in descending order by their score, content is selected to fill the cache until its size threshold. When content is not already available locally, decisions are made towards deciding **how** to copy it from its nearest replica and **when** that operation should be performed. The first decision is based on a simple load balancing strategy, considering the available links' status information and giving priority to direct links, e.g. 3GPP-defined X2 interfaces between eNBs. The second decision is derived from the available time for migration (given by mobility prediction) and the existing schedule for other migrations using the same components. Based on the time it will take to copy the content subset and the deadline to have it copied, a slot is picked in

the migration schedule and the FMC Controller instructs the NDN router at the destination to fetch the content accordingly.

IV. Evaluation Experiments

In order to evaluate the proposed strategies, a set of experiments was defined and is described in detail in the next subsections.

A. Mobility Data Input

In order to evaluate the proposal described in the previous section, a realistic mobility trace was selected [29]. This trace includes data from one hundred human subjects over the course of nine months, and it was collected by MIT students using Nokia 6600 smart phones in the academic year of 2004/2005. Although the information collected includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status, for this evaluation only the information on mobility was considered: Object Identifier (OID), endtime, starttime, person_OID, celltower_OID. Therefore, it is possible to know at every given time to which cell a given person is connected. Such trace can thus be used to assess how the system behaves in realistic conditions, as if mobility was generated in any other way, it would probably create biased results and render the conclusions invalid for real-world scenarios.

Concerning mobility prediction, it is not the main focus of this work. Therefore, 15 minutes delta time predictions (15 minutes in the future) were generated at every 1 hour of simulation time according to the results obtained by mobility prediction works [30][31]. As concluded in the mentioned works, a 50% user movement randomness corresponds to an accuracy of about 50%. Thus, the generated predictions for these experiments had an accuracy following a normal distribution $N(0.5, 0.1)$.

B. Basic Setup

The setup for this evaluation is depicted in Fig. 2. It consists of the proposed architecture implemented in the ns3 simulator using its LTE module [32] together with ndnSIM 2.1 [33]. First, the simulator creates a Content Producer attached to a NDN Router, which is a node with NDN capabilities such as caching and forwarding. The latter is by itself attached using IP and 10 Gbps links to the EPC of the LTE module. Afterwards, a pair of eNB + NDN Router (including a NDN Content Store, i.e. cache of 2 GB stored in RAM) is created for each cell of the trace mobility file, attaching randomly positioned (within the cell's coverage) UEs + NDN Consumers to the LTE network according to the trace mobility inputs. These attachments are changed over the simulation time, thus emulating user mobility and triggering an handover using the X2 interface. That handover is managed by the MME, which is modified to feed information to the Mobility Predictor. The Mobility Predictor feeds mobility information, while NDN Routers provide the remaining relevant information (criteria) to the FMC Controller, which makes decisions and therefore instructs content to be copied between NDN Routers.

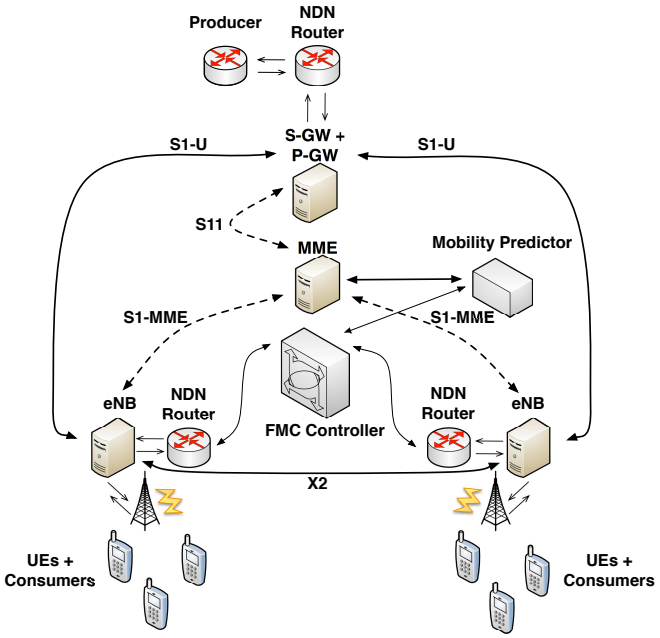


Fig. 2: Evaluation Setup

As for the simulation itself, it runs for 12 hours in a daytime period of the trace mobility file, when it is more likely for users to be active. Simulations were repeated 30 times using different day periods with at least 60 active users and considering the 100 most visited cells, using a Linux cluster to parallelize the work (http://www.ubelix.unibe.ch). Additionally, for parameters not mentioned here, the default values were used (e.g. LTE radio parameters).

C. Decision Criteria's Weights

Assuming that different weights for the criteria may return different results, four different weight sets were considered for evaluation. These are highlighted in the tables below, which each contain the AHP judgment matrix for all the criteria and the resulting eigenvector with weights calculated using the process described in Section III.

	M.P.	Diff	F.S.	G.S.
M.P.	1/1	2/1	4/1	3/1
N.C.	1/2	1/1	2/1	3/1
F.S.	1/4	1/2	1/1	1/2
G.S.	1/3	1/3	2/1	1/1

(a) Matrix

	Weight
M.P.	0.46124
N.C.	0.28450
F.S.	0.10633
G.S.	0.14793

(b) Vector

TABLE I: Weight Set #1

	M.P.	Diff	F.S.	G.S.	Cost
M.P.	1/1	2/1	4/1	3/1	3/1
N.C.	1/2	1/1	2/1	3/1	2/1
F.S.	1/4	1/2	1/1	1/2	1/1
G.S.	1/3	1/3	2/1	1/1	2/1
Cost	1/3	1/2	1/1	1/2	1/1

(a) Matrix

	Weight
M.P.	0.39778
N.C.	0.25232
F.S.	0.09725
G.S.	0.14897
Cost	0.10368

(b) Vector

TABLE II: Weight Set #1 with Cost

In Table I, the importance given to mobility prediction (M.P.) is higher than for any other criterion. At the same time, group size (G.S.) is considered more important than free space (F.S.) and cost is not considered.

	M.P.	Diff	F.S.	G.S.
M.P.	1/1	1/2	3/1	2/1
N.C.	2/1	1/1	3/1	4/1
F.S.	1/3	1/3	1/1	2/1
G.S.	1/2	1/4	1/2	1/1

(a) Matrix

	Weight
M.P.	0.28450
N.C.	0.46124
F.S.	0.14793
G.S.	0.10633

(b) Vector

TABLE III: Weight Set #2

	M.P.	Diff	F.S.	G.S.	Cost
M.P.	1/1	1/2	3/1	2/1	2/1
N.C.	2/1	1/1	3/1	4/1	3/1
F.S.	1/3	1/3	1/1	2/1	1/1
G.S.	1/2	1/4	1/2	1/1	2/1
Cost	1/2	1/3	1/1	1/2	1/1

(a) Matrix

	Weight
M.P.	0.31739
N.C.	0.36398
F.S.	0.11526
G.S.	0.10714
Cost	0.09623

(b) Vector

TABLE IV: Weight Set #2 with Cost

In Table II, everything is similar to Table I besides the fact that migration cost is now considered and the resulting weights are different.

In Table III, the greatest importance is given to the amount of non-intersecting content between the caches (N.C.) and, unlike in Table I, F.S. is considered more important than G.S. Also here, cost is not considered.

In Table IV, everything is similar to Table III besides the fact that migration cost is now considered and the resulting weights are different.

D. Content and Requests

The Content Producer consists of a file generator, which generates 100 000 files according to the defined scenario: either a YouTube scenario or a web server scenario. The first scenario intends to mimic video streaming traffic using conditions from the well-known YouTube video portal, which is the type of traffic that dominates Internet nowadays. The second scenario attempts to mimic traffic of users accessing modern Web 2.0 pages with plenty of multimedia content such as high-resolution images. As shown in Table V, it is assumed that content popularity of both of them follows a Zipf distribution [34]. For this setup, 20 popularity classes are taken into account. As several studies have shown [35][36] that most content objects are unpopular and only a few content objects are very popular, the number of content objects to be included in each popularity class is mapped to a Zipf distribution with $\alpha = 1$ and with inverted classes, i.e., most content objects are included in class 19 and fewest files in class 0.

Parameter	Web Server	YouTube
Requests	Every 5 seconds	
Request Popularity	Zipf distribution with $\alpha = 1$	$\alpha = 2$
File Distribution per Popularity Class	Zipf distribution, $\alpha = 1$ mapped to inverse classes	
File Sizes per Popularity Class	Gamma distribution, $\alpha = 1.8, \beta = 1200$ min. 50KB max. 50MB	Gamma distribution, $\alpha = 1.8, \beta = 5500$ min. 500KB max. 100MB

TABLE V: Evaluation Parameters

As also described in Table V, file sizes within each popularity class are different. Based on existing YouTube models [37], file size distribution for a YouTube scenario is set to a gamma distribution with $\alpha = 1.8$ and $\beta = 5500$. The file sizes for web server traffic are considerably smaller [38]. However, these file sizes have increased during the last years, and it is safe to assume that they keep increasing in the future with NDN. Transmitted NDN packets need to have a certain minimum size to be efficient, e.g., segment size of 4096 bytes or more, to avoid too large overhead for content headers including names and signatures. Therefore, it is assumed that for future NDN traffic, many small files may be aggregated to larger data packets or NDN would only be applied to large static files, e.g., pictures or embedded videos, and not small text files that may change frequently. Therefore, a gamma distribution with $\alpha = 1.8$ and $\beta = 1200$ was selected for the web server scenario.

As for the content requests to be performed by users (Consumers) during the simulation, a parameter of $\alpha = 1$ is considered realistic for web server traffic and $\alpha = 2$ is used for YouTube traffic.

E. Evaluation Metrics

Finally, five different metrics were evaluated to assess accuracy of the strategies, performance improvements for end users and potential savings for operators. The first is the position of an optimal solution (highest profit destination) in the score-sorted ranking of destination alternatives derived from the output of the AHP decision. The optimal solution is the location where the group of users was on which the requested volume of content recently migrated was the highest. If it is in the first positions (1, 2, 3, etc.) of the aforementioned ranking, it means that the decisions were good and will yield benefits for the users. The second metric is the number of cache hits, which enables the comparison of strategies and the benefit to be quantified in terms of end users perspective and possible network bandwidth savings. The third is the average download latency experienced by users, considering the best weight set from previous metrics evaluation. The fourth is the aggregated usage of bandwidth at the core interfaces (S1 and X2), evaluating the overhead caused by different strategies and how the load becomes distributed. Finally, the fifth is a comparison of timings for FMC in the different scenarios in order to evaluate if migrations are made on time when they are reactive (no predictions) or proactive (mobility predicted).

V. Evaluation Results

In the graphs below, results from the experiments defined in the previous section can be observed. To assess the first evaluation metric a comparison is made between the different weight sets, and a Cumulative Distribution Function (CDF) is generated for each ranking position. With the CDF, it is possible to obtain the cumulative percentage of times when the optimal solution was within the n first positions of the ranking. For example, one may assume that x percent of the times the optimal solution was at the first three positions of the ranking of destinations.

In Fig. 3, results show that weight set #1 has a higher percentage of optimal solutions at the first position of the ranking, meaning that selections were perfect in almost 60%

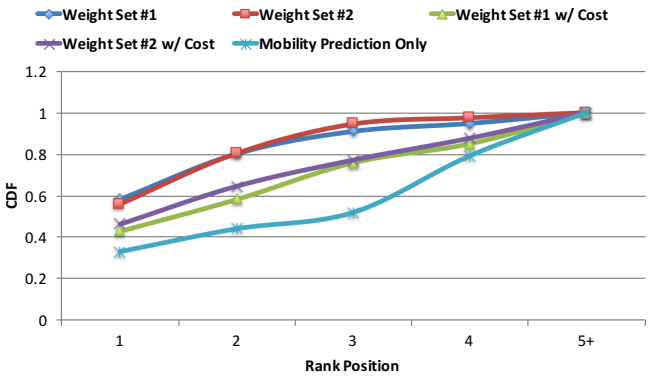


Fig. 3: Optimal Solution in Ranking

of the cases. However, the results of weight set #2 converge quicker to 100% of the cases, surpassing weight set #1 after (and including) rank position number 2. Overall, one may assume that weight set #2 selects better options than any other weight set, especially considering that the optimal solution was within the first three positions of the ranking in more than 90% of the cases. This can be easily explained by the accuracy of mobility prediction, which can vary immensely and does not account for the time users spend at the predicted locations. At the same time, giving priority to destinations where most of cached content is not the same as in the origin has a big inherent potential to be explored from the beginning.

When looking at the weight sets but considering cost, the trend is slightly different. Weight set #2 with cost outperforms weight set #1 with cost from the beginning, with the optimal solution being in the first position of the ranking almost 50% of the cases and in more than 80% of the cases the optimal solution being in the first 4 positions of the ranking. These results are according to what was expected, as cost limits the performance but considering it still delivers a good trade-off for both end users and network operators.

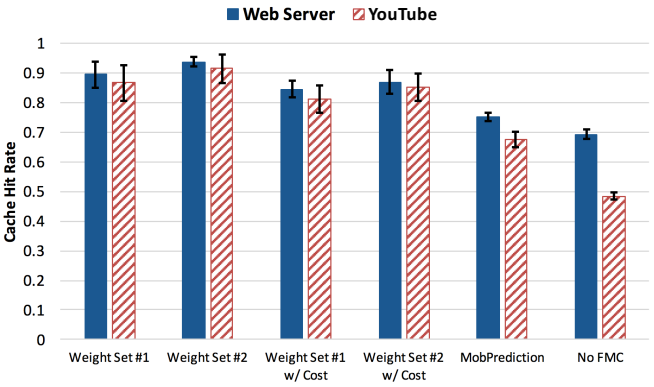


Fig. 4: Cache Hit Rates

As for the second evaluation metric, a comparison between our strategies with different weight sets, mobility prediction only [22] and no use of FMC (default strategy) is shown in Fig. 4, evaluating cache hits in both the YouTube and Web Server scenarios.

From the depicted results, one may observe that cache hit rates tend to be lower for the YouTube scenario because of bigger file sizes and a different Zipf distribution. However, in this particular case our FMC strategies show the biggest difference towards simple Mobility Prediction and No FMC. For instance, the cache hit rate using weight set #2 is up to 40% higher than with the default strategy without FMC, and over 20% higher than relying solely on Mobility Prediction.

As for the Web Server scenario, the benefit is not so high (up to 20% less). Such fact is explained by the characteristics of web server traffic, which has a lot of small objects that are easily cached even if the cache storage space is low. Therefore, users may find most of the content already distributed over the network, and migration strategies do not copy a large amount of content that can yield benefits. However, multimedia content now accounts for the most traffic in mobile networks [1], and we can easily conclude that FMC content migration strategies deliver their biggest performance for the biggest part of the traffic.

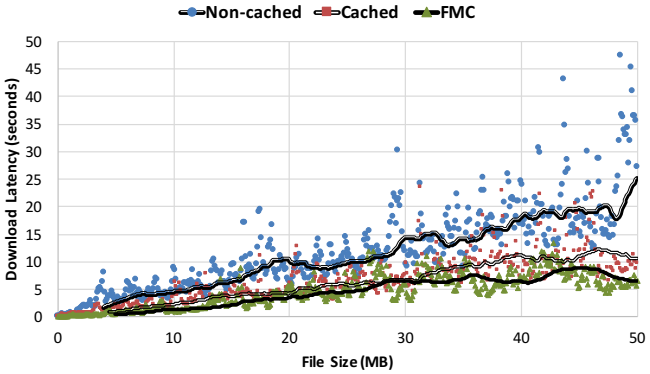


Fig. 5: Average Content Download Latency - Web Server

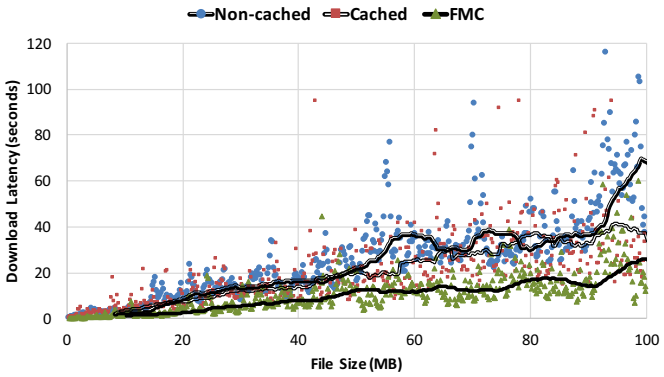


Fig. 6: Average Content Download Latency - YouTube

In Fig. 5 and Fig. 6, a comparison between the different strategies (no edge caching, edge caching and FMC) is presented for the Web Server and YouTube scenarios. Here, the FMC strategy is considered to be Weight Set #2 with cost, thus having the highest cache hit rates together with possible savings in core bandwidth. First, we may observe that data points present high variance, caused by the method they were obtained with. As the number of files is too big to represent, sampling was performed to include only 500 data points between the minimum and maximum file sizes. This sampling considers the sizes of all files generated in the multiple runs, and therefore has the influence of the different file sizes

themselves, network conditions, processing and others. Thus, to facilitate the understanding of the results, a trend line with a moving average of 50 data points is included. Results confirm in a end-to-end user perspective what was visible when comparing cache hit rates: improvements experienced by end users are considerable and caches are used more efficiently, i.e. cached content corresponds mostly to content that will actually be requested by users. This is true for both scenarios, and again we may easily see that improvement towards regular edge caching is much bigger when multimedia traffic is considered and content sizes tend to increase.

As for aggregated core bandwidth usage, Fig. 7 depicts a comparison for the different strategies (no edge caching, edge caching and FMC), in the two scenarios and also in different LTE core interfaces (S1-U and X2). First, we observe that, as expected, the aggregated usage of the S1-U interfaces is clearly reduced for both scenarios when caching at the edge. Second, we can also see that there is an overhead created by using FMC strategies when comparing to edge caching. This overhead becomes more clear over time, when caches start to be filled and more content is migrated, but it is compensated by the usage of the X2 interfaces between eNBs. This balances the load and eventually even adds more load to the X2 interface (prioritize), thus moving traffic away from the EPC and using available resources more efficiently. Finally, we conclude that there are differences between scenarios, especially because of the file sizes being bigger in YouTube traffic. This leads to a higher bandwidth consumption reduction with edge caching in the YouTube scenario, but also a slightly higher overhead for FMC strategies. Despite that, reductions in the usage of S1-U interfaces, and therefore in EPC, are still meaningful because of more traffic being offloaded using X2 interfaces.

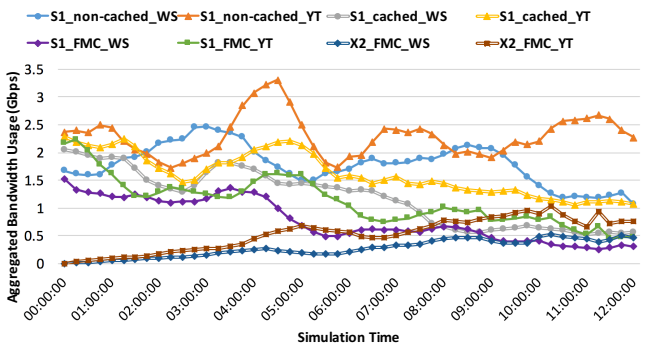


Fig. 7: Aggregated Core Bandwidth Usage

Finally, in Fig. 8 we plot the average execution times of different components for the FMC strategies (in both traffic scenarios) together with the average available time brought by both X2 handover procedures and mobility prediction. All values have a confidence interval of 95%, and we see that despite the accuracy of mobility prediction (about 50%), on average there is still plenty of available time for decisions and other cache operations (bear in mind that we are using a logarithmic scale). Using mobility prediction, in both scenarios the FMC components are able to execute within the available time frame. At the same time, if FMC operations are triggered

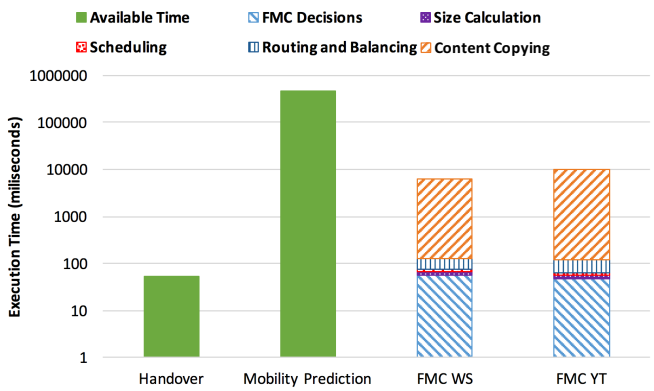


Fig. 8: Execution Times

reactively (no prediction), the handover delay is only enough for the decisions process. This means that the content transfer procedure will only start after the user is already at the new location, and in the worst case scenario it will have some initial cache misses while the content is still transferring. Overall, the impact of this behavior is not very high, as from previous figures we still observed very high cache hit ratios.

VI. Conclusions

In this paper, concepts and strategies for the migration of content within mobile networks were introduced, enabling multiple benefits both from user and network perspectives. As the users move to different locations, they still want to access content in which they are interested with a low latency and without delays or breaks, especially if dealing with multimedia content. From the network perspective, this can only be granted if caches exist at the edge of mobile networks and content kept in those caches (with limited resources) is the right content, i.e. popular content that local users are very interested in.

A number of proposals to handle this issue already exist, and were described thoroughly in Section II. However, some cannot be applied to content (only to services) or have other limitations, often assuming a very specific scenario or scope and not handling important issues or considering certain requirements. Therefore, we propose a broader approach to deal with content migration, handling decisions with multiple criteria and deciding multiple factors that will trigger content migration to a particular place of a given subset of content.

This proposal was evaluated in terms of performance, considering multiple weight values and different scenarios. When comparing to the case where default NDN caching strategies are used, clear benefits can be observed and quantified, leading to the conclusion that not only FMC enhanced caching strategies are the way to go when handling edge caches, but also that the architecture proposed in subsection III-A together with its decision mechanisms can achieve the goal of delivering content with lower latency to end users while efficiently using and saving well-valued network bandwidth.

Although the results can be considered as quite good, improvements can still be made. For instance, more hierarchical levels can be considered in the criteria for the decision where

to scale content and more advanced strategies can be used to decide which subset of content should be migrated. We envision that popularity may not be the only factor to decide which content to migrate due to is general nature, but also other factors that relate user to content should be considered in future work.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union's Seventh Framework Programme Mobile Cloud Networking project (FP7-ICT-318109).

REFERENCES

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf, Feb 2015.

[2] P. Demestichas, A. Georgakopoulos, D. Karvounas, K. Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong, and J. Yao, "5g on the horizon: Key challenges for the radio-access network," *Vehicular Technology Magazine, IEEE*, vol. 8, no. 3, pp. 47–53, Sept 2013.

[3] "Suggestions on Potential Solutions to C-RAN by NGMN Alliance," The Next Generation Mobile Networks (NGMN) Alliance, Tech. Rep., Jan. 2013. [Online]. Available: http://www.ngmn.org/uploads/media/NGMN_CRAN_Suggestions_on_Potential_Solutions_to_CRAN.pdf

[4] B. Haberland, F. Derakhshan, H. Grob-Lipski, R. Klotsche, W. Rehm, P. Schefczik, and M. Soellner, "Radio Base Stations in the Cloud," *Bell Labs Technical Journal*, vol. 18, no. 1, pp. 129–152, 2013.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12.

[6] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 147–158.

[7] D.-h. Kim, J.-h. Kim, Y.-s. Kim, H.-s. Yoon, and I. Yeom, "Mobility Support in Content Centric Networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 13–18.

[8] D. Smetters and V. Jacobson, "Securing network content," PARC, Tech. Rep., Oct. 2009. [Online]. Available: https://www.parc.com/content/attachments/TR-2009-01.pdf

[9] A. Gomes and T. Braun, "Feasibility of Information-Centric Networking Integration into LTE Mobile Networks," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15. ACM, April 2015, pp. 628–634.

[10] X. Wang, M. Chen, Z. Han, D. Wu, and T. Kwon, "Toss: Traffic offloading by social network service-based opportunistic sharing in mobile social networks," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 2346–2354.

[11] M. P. Wittie, V. Pejovic, L. Deek, K. C. Almeroth, and B. Y. Zhao, "Exploiting locality of interest in online social networks," in *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 25:1–25:12. [Online]. Available: http://doi.acm.org/10.1145/1921168.1921201

[12] M. D. Choudhury, H. Sundaram, A. John, D. D. Seligmann, and A. Kelliher, ""birds of a feather": Does user homophily impact information diffusion in social media?" *CoRR*, vol. abs/1006.1702, 2010.

[13] H. Li and G. Ascheid, "Mobility prediction based on graphical model learning," in *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, Sept 2012, pp. 1–5.

[14] S. Rajagopal, N. Srinivasan, R. Narayan, and X. Petit, "Gps based predictive resource allocation in cellular networks," in *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, 2002, pp. 229–234.

[15] Y. Chon, H. Shin, E. Talipov, and H. Cha, "Evaluating mobility models for temporal prediction with high-granularity mobility data," in *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, March 2012, pp. 206–212.

[16] A.-F. Antonescu, A. Gomes, P. Robinson, and T. Braun, "Sla-driven predictive orchestration for distributed cloud-based mobile services," in *Communications Workshops (ICC), 2013 IEEE International Conference on*, June 2013, pp. 738–743.

[17] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *Network, IEEE*, vol. 27, no. 5, pp. 12–19, September 2013.

[18] ——, "An analytical model for follow me cloud," in *Global Communications Conference (GLOBECOM), 2013 IEEE*, Dec 2013, pp. 1291–1296.

[19] A. Ksentini, T. Taleb, and M. Chen, "A markov decision process-based service migration procedure for follow me cloud," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 1350–1354.

[20] H. Liu, Y. Sun, and M. S. Kim, "Provider-level content migration strategies in p2p-based media distribution networks," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, Jan 2011, pp. 337–341.

[21] C. Anastasiades, A. Gomes, R. Gadow, and T. Braun, "Persistent caching in information-centric networks," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, Oct 2015, pp. 64–72.

[22] X. Vasilakos, V. A. Siris, G. C. Polyzos, and M. Pomonis, "Proactive selective neighbor caching for enhancing mobility support in information-centric networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 61–66.

[23] "Named Data Networking (NDN) project," http://named-data.net/techreport/TR001ndn-proj.pdf, PARC, Tech. Rep., Oct. 2010.

[24] D.-L. Xu, "An introduction and survey of the evidential reasoning approach for multiple criteria decision analysis," *Annals of Operations Research*, vol. 195, no. 1, pp. 163–187, 2012.

[25] T. L. Saaty, *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. New York, NY: Mcgraw-Hill, 1980.

[26] R. Saaty, "The analytic hierarchy process—what it is and how it is used," *Mathematical Modelling*, vol. 9, no. 3–5, pp. 161 – 176, 1987.

[27] T. L. Saaty, "How to make a decision: The analytic hierarchy process," *European Journal of Operational Research*, vol. 48, no. 1, pp. 9 – 26, 1990, decision making by the analytic hierarchy process: Theory and applications.

[28] R. Andonov, V. Poirriez, and S. Rajopadhye, "Unbounded knapsack problem: Dynamic programming revisited," *European Journal of Operational Research*, vol. 123, no. 2, pp. 394 – 407, 2000.

[29] N. Eagle and A. (Sandy) Pentland, "Reality mining: Sensing complex social systems," *Personal Ubiquitous Comput.*, vol. 10, no. 4, pp. 255–268, Mar. 2006.

[30] N. Samaan and A. Karmouch, "A mobility prediction architecture based on contextual knowledge and spatial conceptual maps," *Mobile Computing, IEEE Transactions on*, vol. 4, no. 6, pp. 537–551, Nov 2005.

[31] N. Amirrudin, S. Ariffin, N. Malik, and N. Ghazali, "User's mobility history-based mobility prediction in lte femtocells network," in *RF and Microwave Conference (RFM), 2013 IEEE International*, Dec 2013, pp. 105–110.

[32] "ns-3: LTE Module," https://www.nsnam.org/docs/models/html/lte.html, Sep. 2015.

[33] A. Afanasyev, S. Mastorakis, I. Moiseenko, and L. Zhang, "NS-3 based Named Data Networking (NDN) simulator," http://ndnsim.net, Sep. 2015.

[34] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," http://perso.telecom-paristech.fr/~drossi/paper/rossi11ccn-techrep1.pdf, Telecom ParisTech, Tech. Rep., 2011.

[35] T. Yu, C. Tian, H. Jiang, and W. Liu, "Measurements and analysis of an unconstrained user generated content system," in *Communications (ICC), 2011 IEEE International Conference on*, June 2011, pp. 1–5.

[36] "Half of youtube videos get fewer than 500 views," http://www.businessinsider.com/chart-of-the-day-youtube-videos-by-views-2009-5?IR=T, May 2009.

[37] A. Abhari and M. Soraya, "Workload generation for youtube," *Multimedia Tools Appl.*, vol. 46, no. 1, pp. 91–118, Jan. 2010.

[38] A. Williams, M. Arlitt, C. Williamson, and K. Barker, "Web workload characterization: Ten years later," in *Web Content Delivery*, ser. Web Information Systems Engineering and Internet Technologies Book Series, X. Tang, J. Xu, and S. T. Chanson, Eds. Springer US, 2005, vol. 2, pp. 3–21.

Optimized Cooperative Streaming in Wireless Mesh Networks

Luca Baldesi, Leonardo Maccari, Renato Lo Cigno
Department of Information Engineering and Computer Science (DISI), University of Trento, Italy
{luca.baldesi, leonardo.maccari, renato.locigno}@disi.unitn.it

***Abstract*—Peer-to-peer video streaming is a valuable technique to reduce the overhead produced by centralized and unicast-based video streaming. Key to the efficiency of a peer-to-peer approach is the optimization of the logical distribution topology (the *overlay* with respect to the underlying network, the *underlay*). This work studies peer-to-peer streaming in wireless mesh networks for which the underlay is known. We propose an optimized, cross-layer approach to build the peer-to-peer distribution overlay minimizing the impact on the underlay. We design an optimal strategy, which is proven to be NP-complete, and thus not solvable with a distributed, light weight protocol. The optimal strategy is relaxed exploiting the knowledge of the betweenness centrality of the underlay nodes, obtaining two easily implementable solutions applicable to any link-state routing protocol. Simulation and emulation results (experimenting with real applications on a network emulated with the Mininet framework) support the theoretical findings, showing that the relaxed implementations are reasonably close to the optimal solution, and provide vast gains compared to the traditional overlay topology based on Erdös-Rényi models that a peer-to-peer application would build.**

I. Introduction

Video streaming is the major component of the global Internet traffic, and it is thought to be increasing for several years to come. Today streaming is normally delivered with several seconds of delay, but users are asking for more performing systems, and video calls and conferences, which are still somewhat "rare", require a much more timely delivery of the video. Currently, the majority of the Video Service Providers (VSPs) deliver video streams using unicast traffic and leveraging centralized platforms supported by world-wide Content Delivery Networks (CDNs).

Ten years ago, the peer-to-peer (P2P) paradigm and technologies promised to offer a solution for massive content distribution, including video streaming and conferencing. The reasons P2P systems were unable to meet their potential are many, a key one was the difficulty to realize P2P overlays optimized from the point of view of the Internet Service Providers (ISPs).

An application field in which this problem can be overcome is the field of wireless mesh networks, and in particular Community Networks (CN from now on). CNs are large mesh networks (primarily made of wireless links) that are flourishing in many different scenarios, from the developing country where there is no other connectivity means, to the urban areas of western cities where they compete with other network providers. The steep decrease of the prices of outdoor wireless equipment makes it possible to build cooperative wireless mesh networks with links that can achieve tens of Mbit/s and support CNs made of hundreds of nodes. The most prominent example is the Guifi network (www.guifi.net) that is a collection of various networks in East Spain, for a total of about 30,000 nodes [1]. In CNs the underlay is normally known, since the routing protocols exports it to each node (as long as a link-state routing protocol is used), which removes one of the technical barriers that blocked the deployment of P2P video streaming on the Internet. We assume that the distribution engine of the video is installed in the mesh nodes, as already proven feasible in [2], while the video can be enjoyed on standard terminals, so that underlay details are easily accessible to the overlay manager.

The contribution of this paper is a cross-layer optimization scheme to perform *live* video streaming (i.e., with a strict deadline on the arrival delay) in mesh networks [3]. The optimization minimizes the impact of the streaming overlay on the underlay network exploiting information on the topology and routing of the underlay. The case of non real-time streaming can be seen as a sub-case of live streaming relaxing the constraints on delivery delay.

The optimization is based on the concept of *centrality*, which is also at the base of successful algorithms as Google PageRank [4]. Taking into account the centrality of peers in the underlay graph, the optimized overlay topology greatly improves the efficiency of the video distribution and maintains high performance. The resulting algorithm and protocol are first tested on synthetic topologies, showing that they are robust and efficient; finally they are implemented within PeerStreamer[1], an existent live P2P-streaming platform, and tested using a network emulator configured with the topologies of real wireless mesh networks.

II. Motivation and Problem Statement

We consider a communication system where the cooperative distribution exploits a logical topology called the *overlay* built on top of a meshed routing network called the *underlay*.

This work was financed partially by the University of Trento under the grant "Wireless Community Net-works: A Novel Techno-Legal Approach" —Strategic Projects 2014, and partially by the European Commission, H2020-ICT-2015 Programme, Grant Number 688768 'netCommons' (Network Infrastructure as Commons).

ISBN 978-3-901882-83-8

[1]PeerStreamer is Open Source effort supported by the DISI-ANS research group of the University of Trento. See http://peerstreamer.org

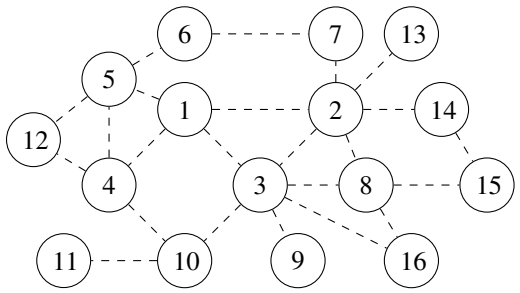


Fig. 1: Example of wireless mesh underlay graph. Hosts are numbered with an arbitrary ordering. Dashed lines represent the wireless links.

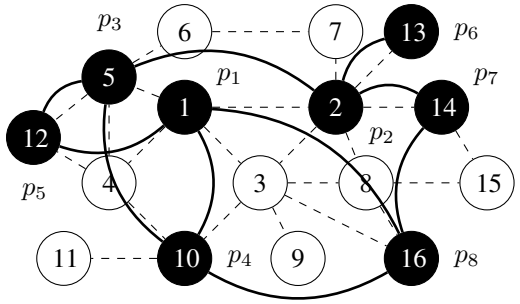


Fig. 2: Possible overlay graph (black vertexes and edges) over the underlay graph of Fig. 1.

Nodes in the *overlay*, that from now on we call peers, do have access to information concerning the *underlay*, including details on the topology and quality of its links. This is true for a wireless mesh network using link-state routing protocols, in which every wireless router needs to know the whole topology of the network to perform routing. For instance, standard implementations as the OLSRd daemon implementing the Optimized Link State Routing (OLSR) protocol [5] export the topology with a simple API. The optimization we propose regards the choices of the edges in the *overlay* so that the impact on the *underlay* is minimized and evenly distributed.

We model the *underlay* with an undirected graph $U(H, L)$ with vertexes $h \in H$ called hosts or nodes, and edges $l \in L$ called links. The size of H is between a few tens up to a thousand hosts, which corresponds to the realistic size of a CN [6]. Fig. 1 shows the graph representation of a sample underlay with 16 hosts.

The peers form an *overlay* that is also modeled as an undirected graph $O(P, E)$ with vertexes $p \in P$ called peers, and edges $e \in E$ called virtual or logical links. Each peer resides in one host only, and it has access to information pertaining to U, including the association between peers and hosts. Fig. 2 depicts a possible overlay graph on the underlay of Fig. 1.

The goal is to find a viable (meaning that can be implemented as a distributed system with limited signaling overhead and acceptable computational overhead) methodology to select virtual links between peers to build $O(P, E)$ given $U(H, L)$ and P so that the load imposed by the video streaming on the *underlay* links is minimized, and links are loaded as fairly as possible. O is dynamically created and maintained because both O and U can change frequently, so the modification of O must be fast and efficient.

TABLE I: Summary of the main symbols used through the paper and their meaning.

Peers, Hosts, Links and Virtual Links sets	P, H, L, and E
Overlay and underlay graphs	$U(H, L)$, $O(P, E)$
Fairness of $O(P, E)$ over $U(H, L)$	$\mathcal{F}$
Network load of $O(P, E)$ over $U(H, L)$	$\mathcal{L}$
kth undirected overlay edge in $P \times P$	e_k
Cross-layer overlay edge descriptor of e_k	$\bar{e}_k$
ith overlay peer	p_i
Set of the overlay edges in $O(P, E)$ linking p_i	S'_i
Family of all S'_i, $i = 1, \ldots, \lvert P \rvert$	F'
$O(P, E)$ as an intersection graph	$\Omega(F')$
Target function on $\Omega(F')$ addressing $\mathcal{L}$ and $\mathcal{F}$	O_c
Binary variable representing of whether $e_k \in E$	z_k
Estimation of link usage in $O(P, E)$	b

A. Formal Problem Definition

Tab. I reports the main notation we use in the paper. Given an edge e connecting $p_i, p_j \in P$, we call D(e) the Dijkstra function returning an (ordered) set of links in the underlay that form the shortest path from the host where p_i resides and the host where p_j resides. For example in Fig. 2 we have:

$$e = (p_1, p_4) \mapsto \mathrm{D}(e) = \{(1, 3), (3, 10)\}$$

where (i, j) is the link e connecting hosts i and j. A generic weight $w(l)$ is assigned to any link in the *underlay*, so that the load $\mathcal{L}$ imposed by O on U is

$$\mathcal{L} = \sum_{e \in E} \sum_{l \in \mathrm{D}(e)} w(l) \tag{1}$$

This representation perfectly fits the routing protocols that use the ETX metric [7], ETX is the expected average number of frames sent on the link to correctly deliver one frame, and it is used by OLSR and other protocols.

Every link l is loaded by a number of virtual links. To measure fairness, we use Jain's fairness index on the distribution of the number of logical links insisting on every l. Let $\mathcal{H}(l)$ be the number of logical links loading l:

$$\mathcal{H}(l) = \lvert \{ e \in E \ : l \in \mathrm{D}(e) \} \rvert \tag{2}$$

where $\lvert \cdot \rvert$ is the size of a set. The Jain's fairness is defined as

$$\mathcal{F} = \frac{\left(\sum_{l \in L} \mathcal{H}(l) \right)^2}{\lvert L \rvert \sum_{l \in L} \mathcal{H}(l)^2} \tag{3}$$

Jain's fairness is maximal if $\mathcal{F} = 1$ and minimal when $\mathcal{F} = \frac{1}{\lvert L \rvert}$, but we do not expect that maximal fairness can be reached, as in general there are links in the *underlay* that do not support any edge in the *overlay*.

The contribution of this paper is twofold:

- First we derive a formal framework wherein it is possible to define an optimization problem that allows finding the topology of $O(P, E)$ given $U(H, L)$ and P that minimizes a metric composed of $\mathcal{L}$ and $\mathcal{F}$. The problem is NP-complete and we'll show that it can be reduced to a quadratic knapsack problem [8];

- Second we propose two relaxations of the optimization in decreasing complexity order and we show, with numerical solutions and with an implementation in a real P2P streaming platform, that the two relaxations are close to the global optimum and that they vastly outperform the tradition P2P *overlay* building based on selecting uniformly neighboring peers to build an Erdös-Rényi graph.

The paper focuses on building a mesh overlay and not on how video chunks are distributed on it. On mesh topologies this latter problem can be tackled with several different strategies. Even if this is not the focus of the paper, Sec. III briefly refers some relevant works on this topic, justifying our choice for the distribution strategy we use in Sec. VI and Sec. VII. The assumption we start from, which becomes a constraint of the optimization problem, is that each peer should receive exactly one copy of every chunk. We assume also that each peer will contribute to the dissemination serving chunks to other peers proportionally to its degree in O, so that the chunk dissemination strategy becomes agnostic of the volume of served chunks, simplifying it. This is one reason why the *overlay* cannot simply be a full mesh, since the traffic generated by each node is proportional to its degree. Another reason is that to maintain a set of neighbors each peer will periodically send probe messages to verify their state and possibly the network conditions (loss, delay etc.). Again the overhead needed for the maintenance of the network graph is linear with the degree of each node. The theoretic findings are validated implementing the strategies devised in Sec. V into the PeerStreamer platform [9], and comparing them with the standard overlay management strategy available in PeerStreamer and used as reference for several works like [10], [11].

III. Related Works

Cooperative video streaming (including P2P) is an established research area. We focus on unstructured and mesh-based approaches, in which there is no specific structure (like a tree) in the topology. This approach has been shown to be particularly robust even in networks with churn (i.e., peers leaving and joining the swarm).

We do not consider here papers that perform streaming optimization on mesh networks requiring modifications to the lower layers (they cannot be applied to existing CNs) or that are not tailored for live video streaming (e.g., using large chunks that imply several seconds of buffering delay). We also do not consider techniques (e.g., like cloud-assisted or SDN based), where the role of the peers is, in one way or another, not fundamental, and we assume that security [12] and collusion [13] issue need not be solved by the application itself. The following discussion is focused on two parts: *topology management*, which is directly related to our contribution, and *chunk/information scheduling*, which justify the choice of the chunk selection strategy in Sec. VI and Sec. VII.

1) Topology Management: As we already mentioned, *overlay* optimization on the Internet is not feasible due to lack of information on the *underlay* details; however several efforts have been done to adapt and improve the *overlay* topology to some measured *underlay* characteristics.

The first approach to mention is the use of "network coordinates" as a means to compute distances between hosts in a certain space. Several algorithms were proposed [14], [15], [16], that are designed to work in the heterogeneous environment of Internet. In all of them the goal is clearly to find a method to infer details on the *underlay* (the Internet), a problem that we do not have, as we take advantage of the available information on the network topology provided by routing protocols in Wireless Mesh Networks (WMNs). We believe that our solution can be adapted, albeit not straightforwardly, to situations where the *underlay* is not known but approximated with network coordinates.

A second line of research has been concerned with the adaptation of the *overlay* based on bandwidth [17] or delay (normally the round trip time between peers) [18] measures, but also on a mix of the two [19]. The solutions found in these works are, once again, tailored to the Internet, where delays can be large (CNs spans a few tens of hundreds of km at most), and bandwidth asymmetry at the edges impose hard limits to the capacity of peers to contribute to dissemination.

Extremely interesting and promising for topology management is the adoption of centrality metrics as means to better understand the topology characteristics of a network graph as it emerges from the routing protocol. Centrality metrics in graphs have been used in social science since the 70s to identify the most influential elements in social networks. Quite surprisingly, they were not applied to multi-hop networks up to recent times [4]. Centrality metrics can be used to enhance network monitoring and routing [20], intrusion detection and firewalling [21], [22], and topology control [23]. There are several metrics based on different centrality "concepts". In this paper we use the betweenness centrality (see [22] for a definition tailored to our problem) to relax the optimization problem as it is strictly related to shortest path routing.

2) Chunk Scheduling: In a mesh-based *overlay* the problem of chunk scheduling is the selection of the neighbors to send/receive information to/from while contextually choosing the right information (chunk) to send/retrieve. This problem has been extensively studied [24], [25], [26], and in some specific contexts with restrictive assumptions, the existence of an optimal scheduling strategy has been proven [27].

Those works show that an efficient and robust chunk scheduling technique that works well in most environments consists in selecting a neighbor with a random (possibly weighted) strategy, and push the most recent chunk that is still missing at the receiving peer (Latest Useful Chunk). This is the standard methodology used in PeerStreamer and that we use to benchmark our proposal.

IV. Overlay Model

The links in U are bidirectional and assumed quasi-symmetric: the most common routing protocols take care of excluding unidirectional or highly asymmetric links. Thus both O and U are undirected, and the maximum number of edges they can have is $m_O = \frac{|P|^2-|P|}{2}$ and $m_U = \frac{|H|^2-|H|}{2}$ respectively.

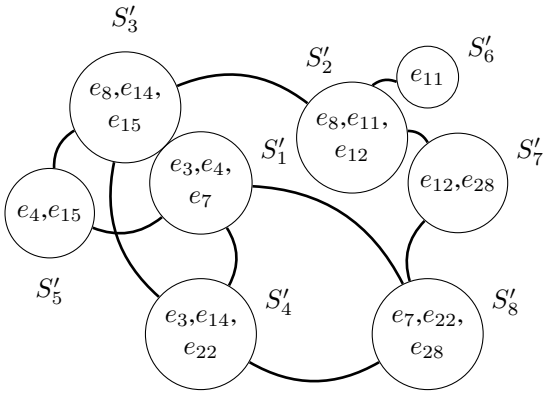


Fig. 3: Example of overlay intersection graph. Elements e_k are the cross-layer overlay edge descriptors between the nodes.

Let $r \in 1 \dots m_U$ be an arbitrary ordering on the links; r is also a mapping from the two hosts h_i and h_j that are the endpoints of the link: $r(h_i, h_j)$. Similarly we define an ordering $k \in 1 \dots m_O$ on the edges of O, and k is a mapping $k(p_i, p_j)$.

Every link l_r is represented by a binary array of size m_U with the rth element set to one and all other elements set to zero (we use the bar sign to refer to the array representation of a link):

$$\bar{l}_r = (0, \dots, 0, 1, 0, \dots, 0)$$

Equivalently each virtual link e_k is represented by the sum of the link arrays in the corresponding shortest path:

$$\bar{e}_k = \sum_{l_r \in \mathrm{D}(e_k)} \bar{l}_r$$

We call $\bar{e}_k$ the cross-layer overlay edge descriptor, since it connects the overlay with the underlay. Link weights $w(l)$ in U can be easily taken into account: let $W \in \mathbb{R}^{m_U \times m_U}$ be a diagonal matrix such that $W_{r,r} = w(l_r)$, then $\bar{e}_k W$ is the weighted representation of the overlay edge.

A. Overlay re-definition as an intersection graph

Given all $\bar{e}_k$ for a set of peers P, we can take advantage from a transformation into the intersection graph space (see [28] for the complete definitions and properties of intersection graphs) in order to formulate our optimization problem.

Let S be a set and $F = \{S_1, \dots, S_p\}$ a nonempty family of distinct nonempty subsets of S whose union is S. The intersection graph of F is denoted $\Omega(F)$, with S_i and S_j adjacent whenever $i \neq j$ and $S_i \cap S_j \neq \emptyset$. It is easily shown that if $O(P, E)$ is a full mesh then it is isomorphic with the intersection graph space $\Omega(F)$ where

$$S_i = \{\bar{e}_{k(p_i,p_j)}, \forall p_j \in P\} \ \forall p_i \in P; \quad S = \cup_{i=1}^{|P|} S_i \qquad (4)$$

and each S_i is the set of all the possible virtual links built on shortest paths from peer p_i to all other peers. As a consequence, given an underlay $U(H, L)$ and a set of peers P, any overlay $O(P, E)$ over $U(H, L)$ can be defined as the intersection graph $\Omega(F')$ with $F' = \{S'_1, ..., S'_{|P|}\}$ where $S'_i \subseteq S_i \ \forall S_i \in F$. If each peer chooses only a subset $S'_i \subseteq S_i$ and activates only a subset of possible edges, then the resulting overlay is isomorphic with some $\Omega(F')$. The overlay depicted in Fig. 2 is isomorphic with the intersection graph shown in Fig. 3: every S'_i in Fig. 3 corresponds to a peer p_i in Fig. 2.

B. Performance Measures

We can now redefine in terms of intersection graphs also the performance metrics Eq. (1) and Eq. (3) defined in Sec. II:

$$\mathcal{L} = O_I(\Omega(F')) = \vec{1} \cdot L(F'); \quad L(F') = \sum_{\bar{e}_k \in S'} \bar{e}_k W \qquad (5)$$

where $L(F')$ is the array that associates the traffic potentially produced by the overlay to each link in the underlay, $\vec{1}$ is the array of size m_U made of all ones, and $\cdot$ is the dot product. Eq. (5) redefines Eq. (1) through operations done in the intersection graph space. Similarly we re-define the Jain's fairness index as

$$\mathcal{F} = O_f(\Omega(F')) = \frac{(\sum_{k=1}^{m_U} L(F')_i)^2}{m_U(\sum_{k=1}^{m_U} L(F')_i^2)} \qquad (6)$$

V. Overlay optimization

For the sake of simplicity, but without loss of generality we take as weighting matrix W the identity matrix. Let's say we want to build an overlay O determined by a choice of $F' = \{S'_1, ..., S'_{|P|}\}$ that minimizes the load on the underlying edges and guarantees a fairness as close as possible to 1. We have to choose the sets $S'_i \subseteq S_i$ with $S' = \cup_{i=1}^{|P|} S'_i$ such that both O_I and O_f are minimal, which is a multi-objective combinatorial optimization problem. The problem allows the definition of a combined metric O_c that expresses the *cost* of the overlay. Since each array $\bar{e}_k \in S'$ corresponds to a set of links the cost of the overlay is defined as:

$$O_c(\Omega(F')) = \left|\left| \sum_{k=1}^{|S'|} \bar{e}_k \right|\right|_2$$

The creation of an efficient overlay $\Omega(F')$ can now be formulated as a minimization problem as follows. Select F' in order to minimize the expression:

$$\arg\min_z \left|\left| \sum_{k=1}^{|S|} z_k \bar{e}_k \right|\right|_2 \qquad (7)$$

where

$$z_k = \begin{cases} 1 & \text{if } \bar{e}_k \in S' \\ 0 & \text{otherwise} \end{cases}$$

In order to avoid a trivial solution we impose on each peer a minimum node degree $d > \log_2(|P|)$, which also guarantees that the resulting overlay is connected with high probability[2].

This problem can be rephrased as: find the overlay graph $\Omega(F')$ with minimum degree d defined by $F' = \{S'_1, ..., S'_{|P|}\}$

[2]Using simple P2P techniques the probability actually converges to 1 [19].

so that the norm of $L(F')$ is minimized subject to the following constraint

$$\left(\sum_{\bar{e}_k \in S_i} z_k\right) \geq d; \quad \forall i = 1...|E| \tag{8}$$

In Eq. (8) we did a small abuse of notation to improve the readability: the sum spans all the edges $e_k \in S_i$, but it is indeed a sum over $k(p_i, p_j)$ to correctly identify the indication function z_k. We will use this notation also in several other equations.

If, instead of the norm, in Eq. (7) we use the sum of the elements, this would simply minimize O_I. The norm instead prefers solutions that are close to the minimum O_I and, among two potential solutions with the same O_I, it prefers the one in which the weights of $L(F')$ are more fairly distributed.

This problem is a zero-one quadratic programming problem [29], similar to a quadratic knapsack problem [8]. In this kind of problems one wants to minimize the value of an expression $c^T x + x^T Q x$ where $x = \{0,1\}^n$ is an array of binary variables, $c \in R^n$ and Q is a symmetric matrix of size $n \times n$. The minimization is subject to a constraint of the kind $h^T x + x^T G x > g$ where h is an array of size n, G a symmetric matrix of size $n \times n$ and g some real value. If we call $\bar{A}$ the matrix made of columns corresponding to the arrays $\bar{e}_k$ and z the array made of z_k elements then:

$$\arg\min_z \left|\left|\sum_{k=1}^{|S|} z_k \bar{e}_k\right|\right|_2 = \arg\min_z z^T \bar{A}^T \bar{A} z$$

Our problem is thus a zero-one quadratic problem with $Q = \bar{A}^T \bar{A}$, G and c made of all zeros, $h = \vec{1}$. This family of problems is known to be NP-hard, but there are algorithms in literature that make them tractable up to a certain size using branch-and-bound techniques. Still, when $|S|$ grows beyond a few hundreds the problem can not be solved on commodity hardware. $|S|$ corresponds to the number of possible edges in the overlay m_O, so it scales quadratically with the number of peers, which quickly makes the problem intractable.

A. Betweenness Centrality-based Relaxation

We need to find a relaxation in which each p_j can solve a portion of the problem, making some assumptions on the behaviour of the other peers. This corresponds to a scenario in which every peer is aware of the other peers, independently selects its own neighbors, and it communicates them its choice. This is the way P2P streaming protocols based on peer sampling typically work (including PeerStreamer).

Let us first separate the contribution to the overall cost of the edges chosen by p_j and all the other peers in Eq. (7)

$$\left|\left|\sum_{k=1}^{|S|} z_k \bar{e}_k\right|\right|_2 = \frac{1}{2}\left|\left|\sum_{S_i \in F, S_i \neq S_j} \sum_{\bar{e}_k \in S_i} z_k \bar{e}_k + \sum_{\bar{e}_k \in S_j} z_k \bar{e}_k\right|\right|_2 \tag{9}$$

The value $\frac{1}{2}$ comes from the observation that when we separately count each link, every link is counted twice in the sum. Let's call b_j the vector representing the choices of the peers in $P \setminus \{p_j\}$:

$$\left|\left|\sum_{k=1}^{|S|} z_k \bar{e}_k\right|\right|_2 = \frac{1}{2}\left|\left|b_j + \sum_{\bar{e}_k \in S_j} z_k \bar{e}_k\right|\right|_2 \tag{10}$$

and we can say that:

$$\arg\min_z \left|\left|\sum_{k=1}^{|S|} z_k \bar{e}_k\right|\right|_2 = \arg\min_z \left|\left|b_j + \sum_{\bar{e}_k \in S_j} z_k \bar{e}_k\right|\right|_2 \tag{11}$$

Our goal is now to find a relaxation of the problem in which every peer chooses its own neighborhood making some assumptions about b_j, which represents the choice that the other peers $p_i \neq p_j$ do. We need to find a reasonable approximation $\bar{b} \simeq b_j$ that node p_j can use in (11).

Let us now introduce a notion that helps us in this task. In graph theory, the notion of *betweenness centrality* is a property of the edges (or nodes) of a graph defined as the fraction of the total number of shortest paths that passes through that edge (or node). It is a metric used to identify the edges (or nodes) that are more involved in multi-hop interactions between the vertexes of a graph, so for some applications they can be considered more important than the others.

We call $b = \sum_{\bar{e}_k \in S} \bar{e}_k$ the summation of all the cross-layer overlay edge descriptors of the complete overlay. Recall that $\bar{e}_k$ corresponds to a shortest path in U between two hosts on which a peer resides, thus, each element of b corresponds to a link in U and expresses the number of shortest paths in the set S that insist on that link.

Consider the limit case in which every host in the underlay contains a peer ($|P| = |H|$) and let b^* be the value of b normalized to the total number of shortest paths:

$$b^* = b\frac{2}{|P|^2 - |P|}$$

b^* is exactly the vector corresponding to the betweenness centrality of each link in L.

If $|H| > |P|$, b^* is an approximation of the real array of centralities. This is a known fact that is used to approximate centrality in large networks: if the number of nodes is too large to compute all the shortest paths, centrality can be estimated using a subset of the paths chosen from a random set of nodes [30]. A key fact is that the convergence to a solution close to the real one is pretty fast in power-low graphs, that are extremely frequent in real communication networks, and also in some large CNs [1]. Thus, even if p_j ignores b_j a reasonable assumption is that whatever the choice of each other peer is, the elements of b_j (that represent the sum of all choices) have a shape similar to the centrality expressed by the normalized value b^*. This, on power-law underlays is true even for values of $|P|$ one order of magnitude smaller than $|H|$.

Given that we impose each peer to have (at least) d neighbors, the number of edges in E will be approximately $\frac{d|P|}{2}$ and finally the best approximation for b_j turns to be

$$\bar{b} = b^*(|P|-1)\frac{d}{2} = b\frac{d(|P|-1)}{|P|^2-|P|} = b\frac{d}{|P|} = \frac{d\sum_{k=1}^{|S|} \bar{e}_k}{|P|} \tag{12}$$

The complexity of Eq. (12) is polynomial with P [30], that allows the computations of its solution for overlays of hundreds of peers using commodity hardware. Moreover in communication networks, that are sparse graph, betweenness can be computed quickly using heuristics [31], which explains why we introduced this formulation based on centrality. Thus, replacing b_j with $\bar{b}$ in (11) each peer p_j resolves the following optimization problem:

$$\arg\min_{z} \left\| \bar{b} + \sum_{\bar{e}_k \in S_j} z_k \bar{e}_k \right\|_2 \tag{13}$$

$$\text{conditioned to: } \sum_{\bar{e}_k \in S_j} z_k \geq d \tag{14}$$

The formulation of (13) is another zero-one quadratic minimization problem, but the dimension of the problem is now bounded by $|P| < |S|$ (the maximum number of neighbors for p_j), and can be effectively solved up to hundreds of peers. In the rest of the paper we will use the branch-and-bound solver given by the YALMLIP library [32] which solves the problem (13) for a network of 100 nodes in few seconds.

If still the dimension of the problem or the available hardware do not allow the solution of the optimization, we can apply a greedy search algorithm, ranking each possible $\bar{e}_k$ for its weight and choosing the ones that minimize the sum. This corresponds to relax (13) to:

$$\arg\min_{z} \sum_{\bar{e}_k \in S_j} z_k \left\| \bar{b} + \bar{e}_k \right\|_2 \tag{15}$$

$$\text{conditioned to: } \sum_{\bar{e}_k \in S_j} z_k \geq d \tag{16}$$

which of course captures only a part of the original problem but greatly simplifies the computation. Our results show that the solutions generated by (7), (13) and (15) in the case of realistic network topologies are reasonably close one another.

It is worth nothing that if we set $b = \vec{0}$ in Eq. (15) then the l-2 norm respects the order of a l-1 norm. In practice each p_j chooses the neighbors that are close in terms of hops in the underlay. The b terms instead introduces a bias in the choice towards the neighbors connected through links that are less overloaded and introduces a higher fairness. In the comparison we include also a strategy in which $b = 0$ because this strategy can be used also in absence of full information from the underlay topology, since the distance from another peer can be measured with probing tool (like the `traceroute` application) or can be inferred by the time-to-live field in IP packets. Results show that the performance of this simple ranking function is sensibly lower compared to the proposed strategies.

Tab. II summarises the different optimization strategies and labels them with names used in the rest of the papers.

Name	Symbol	Formula
Global Optimization	G_o	$\arg\min_z \parallel \sum_{k=1}^{\lvert S\rvert} z_k \bar{e}_k \parallel_2$
Local Optimization	L_o	$\arg\min_z \parallel \bar{b} + \sum_{k=1}^{\lvert S_j\rvert} z_k \bar{e}_k \parallel_2$
Local Equalized Ranking	E_r	$\arg\min_z \sum_{k=1}^{\lvert S_j\rvert} z_k \parallel \bar{b} + \bar{e}_k \parallel_2$
Local Ranking	L_r	$\arg\min_z \sum_{k=1}^{\lvert S_j\rvert} z_k \parallel \bar{e}_k \parallel_2$

TABLE II: A summary of the optimization functions

VI. Experimental set-up

We evaluated the proposed strategies with two different approaches, first, we implemented them in a simulator that computes the best overlay according to each strategy on synthetic network topologies, then we run a modified version of PeerStreamer in an emulated network topology derived from previous studies of real mesh networks.

A. Simulations

Simulations have been performed using the Networkx library, a powerful library for the generation and analysis of graphs realized in the Python language. Given an underlay topology we have implemented the proposed strategies and for each one we compared the measures of load and fairness of the generated overlay graph. The global and the local optimization problems are solved with the YAMLIP library, while the others have been implemented directly in Python.

B. Emulations

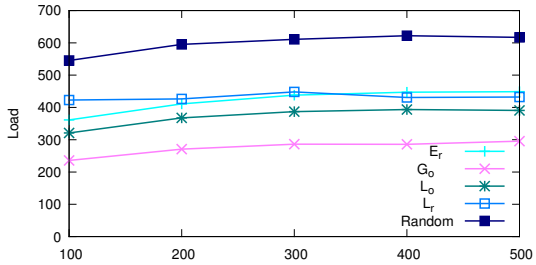
All the code used for emulations is open-source and is freely available on-line[3]. For our emulation experiments we take advantage from real-life WCN topologies, taken from the Ninux and the FFWien[4] networks [33] respectively made of 131 and 236 nodes.

Emulations are based on a modified version of Mininet, a lightweight emulator for arbitrary network topologies [34]. We use the ETX information from the topology dataset to evaluate the link loss during experiments. We consider a link delay uniformly distributed in $[30, 1000]$ µs and a constant link bandwidth of 10 Mbit/s. This is a reasonable assumption since WCN links can typically provide even more bandwidth, both in uplink and downlink. The sample video we use in our experiments is a re-encoding at bit rate of $300\,\mathrm{kbit\,s^{-1}}$ (including both audio and video) of Big Buck Bunny[5]. Currently, hardware constraints limit the number of overlay nodes we can emulate, in our experiments we setup overlays of 30 peers with a minimum node degree of 10.

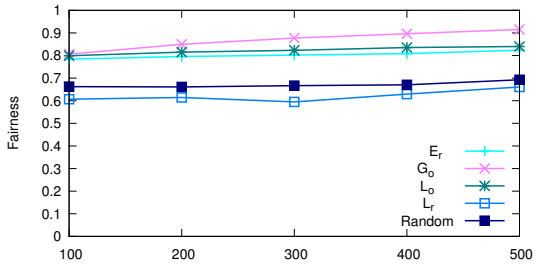
VII. Results

Figs. 4 and 5 report the comparison of all the described strategies in a small (20 peers) scenario, increasing the size of the underlay. With 20 peers we are able to solve all the optimization problems, so this is a good benchmark to outline the differences between each strategy. Figs. 4a and 4b report the differences in the strategies measured using an Erdős-Rényi (ER) underlay of increasing size, while Figs. 5a and 5b report the same values computed on a Barabási-Albert (BA) graph of the same size. In all the cases, the optimized results

[3]https://ans.disi.unitn.it/redmine/projects/peerstreamer
[4]See http://ninux.org and http://www.funkfeuer.at/
[5]http://www.bigbuckbunny.org

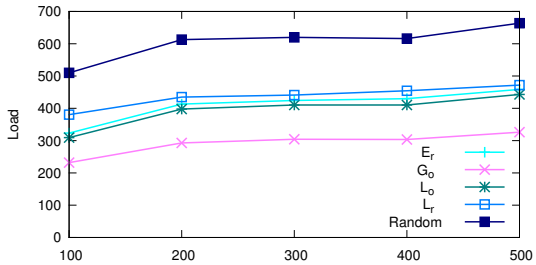


(a) Load for 20 peers

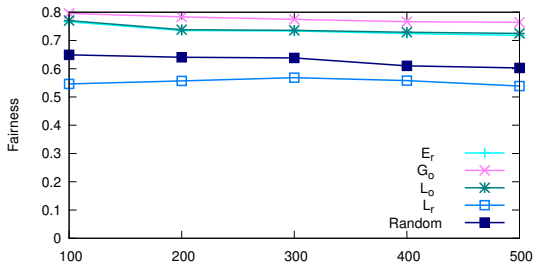


(b) Fairness for 20 peers

Fig. 4: Load and fairness for an ER underlay from 100 to 500 hosts and 20 peers



(a) Load for 20 peers

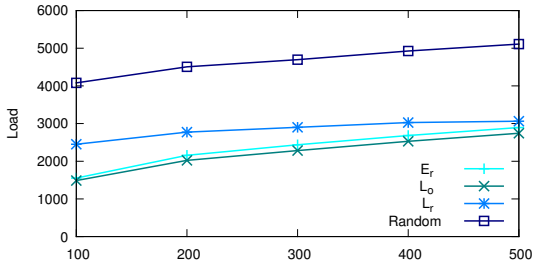


(b) Fairness for 20 peers

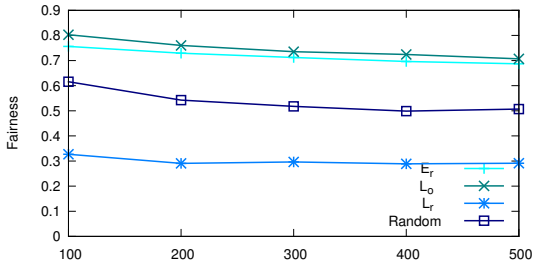
Fig. 5: Load and fairness for a BA underlay from 100 to 500 hosts and 20 peers

vastly outperform the random strategy, which is not surprising since they use available information on the underlay. What is most significant is that even the strategies that are less costly to compute, namely L_o and E_r achieve results that are very close to the optimal strategy G_o especially in terms of fairness. This is true for both the topology types chosen, and more evident for the BA graphs.

This means that even if the quadratic optimization remains NP, in the analysed graphs (especially for BA graph) the



(a) Load for 100 peers



(b) Fairness for 100 peers

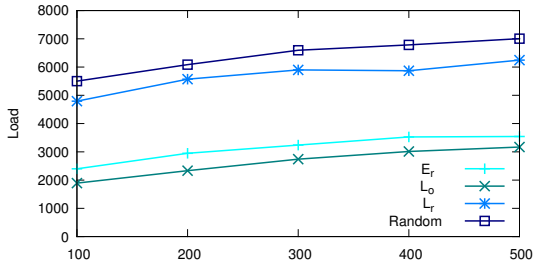
Fig. 6: Load and fairness for a BA underlay from 100 to 500 hosts and 100 peers

number of available disjoint paths between two peers is low and the space of the solutions of the optimization problem is small enough for all the optimization strategies to be very close. This is a key observation since many works in literature show that real communication networks present a scale-free topology as the BA algorithm produce.

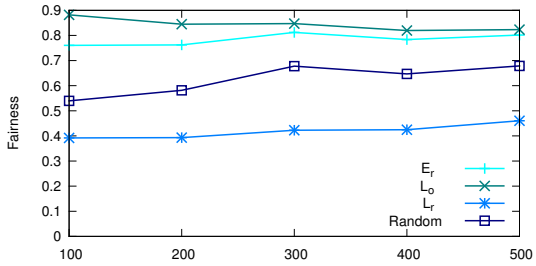
In the next set of results we evaluate the strategies using two different topology generator, the already mentioned BA and the generator proposed by Cerdá-Alabern (CE) in [1], derived from the analysis of a number of real mesh networks. The CE algorithm uses a preferential attachment algorithm to create a core of interconnected hosts, and then adds leaf hosts using a Gamma distribution. We performed experiments with overlay size up to 100 peers, since there is no qualitative difference in the results we report only the results for 100 peers. With that size we are not able to use the G_o strategy, so the comparison is done only with the remaining strategies.

Figures 6 and 7 report the results on a 100 peers overlay and show that in both the considered topologies the L_o and the E_r strategies are very close and achieve a substantial improvement compared to the random strategy.

It is interesting to note that the random strategy generates a distribution of the traffic in the underlay in which the links with a higher centrality have a higher load. In practice, the value of fairness computed on the random overlay mirrors the fairness computed on the b^* array for the underlay. Only the L_r strategy produces a lower fairness compared to the random strategy, since L_r simply chooses the neighbors for peer p_j among the closest ones. It thus reduces the overall load (since the average distance is decreased) but it prefers links that are highly central, and peers that are highly central, thus decreasing the measure of fairness. This observation is important to understand that to achieve a fair distribution of



(a) Load for 100 peers



(b) Fairness for 100 peers

Fig. 7: Load and fairness for a CE underlay from 100 to 500 hosts and 100 peers

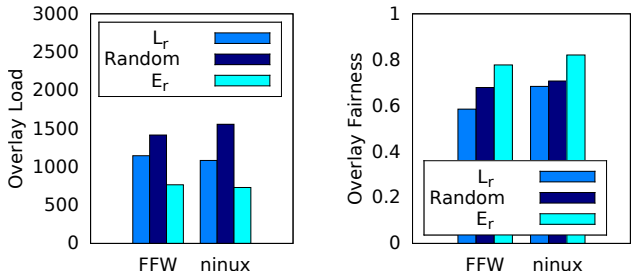


Fig. 8: Load and fairness of the overlay graph computed on FFWien and ninux topologies

the resources it is necessary to have at least some information on the underlay, or else the local-only optimization performed by L_r leads to a global choice that is largely sub-optimal.

A. Results on Real Topologies

To further corroborate our results, first we tested the proposed strategies on two topologies that have been extracted from real mesh networks, then, on the same topologies we run a modified instance of PeerStreamer that implements the E_r and L_r strategies using the emulation environment. The goal of this subsection is to show that the real implementation has even better performance compared to the simulated algorithm, that the E_r strategy can be easily implemented and that it guarantees a timely delivery of chunks in a realistic wireless mesh network scenario.

Fig. 8 shows the load and fairness on the overlay of the ninux and FFWien networks, obtained with simulations. The results are perfectly compatible with the ones we have described so far. Moreover, they are perfectly compatible with Fig. 9 that reports the corresponding values measured with the

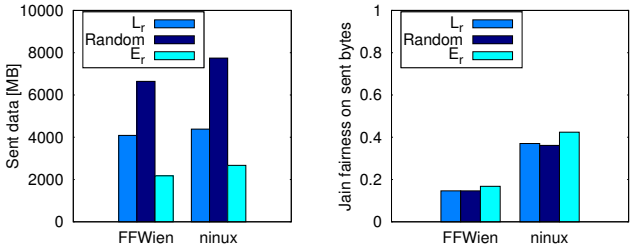


Fig. 9: Sent data and data fairness of the overlay graph measured with PeerStreamer on real topologies

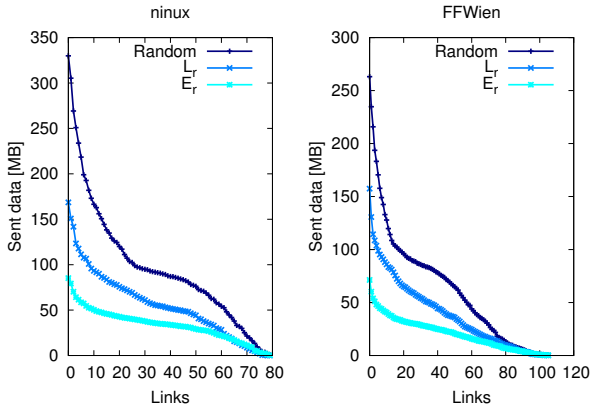


Fig. 10: Load measured on each link on real topologies

PeerStreamer emulation. In this case the occupation of each link has been measured in terms of number of video bytes transmitted on each underlay link. PeerStreamer includes a number of mechanisms that limit the traffic per link, while in the simulations we considered that each link would have carried an unit of traffic. For this reason the absolute values are not comparable, indeed the relative values show an extremely similar behaviour, especially in the load measure. The distribution process operated by PeerStreamer is extremely dynamic and driven both by its algorithms and by random choices. This very dynamic behaviour leads to an hardly predictable link resource usage and it may reduce the absolute value of the fairness.

The graph in Figure 10 show the number of bytes sent on each link of the underlay measured on all the emulation runs. Links are ordered on the x axis for the measured load, reported on the y axis. The graph confirms the decreased total traffic, it shows that L_r and E_r have a small deviation from their average value (which confirms the fairness measure) but it also shows that the peak of the measured traffic is strongly reduced. This is a fundamental feature that shows that a fair distribution of resources can effectively prevent bottlenecks and saturation on the most central links.

Finally, Tab. III reports the packet loss in the emulated network. We do consider successfully delivered only chunks that arrive before 1 second after their generation, chunks arrived with a higher delay would not be useful in a real-time live streaming. It is evident that the proposed strategies do not significantly impact the delivery of packets, which is

Network	Strategy	Loss (%)
FFWien	L_r	4.8
FFWien	E_r	4.9
FFWien	Random	3.8
ninux	L_r	1.5
ninux	E_r	1.6
ninux	Random	1.6

TABLE III: Packet loss measured via emulation

in both networks higher than 95%. Ninux has lower losses since it has a higher average link quality[33]. Note that with a lower delivery rate it would be necessary to assess the received video with specific video-quality metrics, but with less than 5% or even 2% loss, we do not consider necessary to apply such metrics, and we can focus only on networking issues.

VIII. Conclusions

WMNs are an integral part of the present and future Internet access, as they provide large capacity with flexible allocation and mobility. Thus, video streaming in WMNs with a small footprint on the precious wireless resources is of paramount importance. Starting from this observation we designed a novel strategy for cooperative (live) video streaming on distributed networks which can be successfully deployed on a WMN, or any other multi-hop network that provides to the network nodes a complete view of the topology.

Exploiting a novel mapping of the problem onto intersection graphs, we formulated an optimization problem to build an overlay that not only reduces the total load on the underlay, but also increases the fairness in the distribution of the load on the underlay links. This problem in its general formulation is NP, so we proposed two relaxations based on betweenness centrality.

We applied the proposed technique to video streaming on the PeerStreamer platform, but we believe it can be applied to any similar problem in which an overlay must be optimized to efficiently use the resources of the underlay network. Moreover, recently introduced heuristics make centrality metrics fast to be computed even on thousands of nodes so our proposal can be extended to networks and applications with a larger number of peers than what we analysed in the paper.

References

[1] L. Cerda-Alabern, "On the topology characterization of Guifi.net," in *IEEE 8th Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2012, pp. 389–396.

[2] L. Maccari et al., "Live Video Streaming for Community Networks, Experimenting with PeerStreamer on the Ninux Community," in *ACM DIYNetworking '15*, Florence, IT, May 22, 2015, pp. 1–6.

[3] L. Baldesi, L. Maccari, and R. Lo Cigno, "Improving P2P streaming in Wireless Community Networks," *Computer Networks*, vol. 93, Part 2, pp. 389–403, 2015.

[4] D. Katsaros, N. Dimokas, and L. Tassiulas, "Social network analysis concepts in the design of wireless ad hoc network protocols," *IEEE Network*, vol. 24, no. 6, pp. 23 –29, Dec. 2010.

[5] T. Clausen and P. Jaquet, "Optimized link state routing protocol (olsr)," Internet Requests for Comments, RFC 3626, Oct. 2003.

[6] B. Braem et al., "A Case for Research with and on Community Networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 68–73, Jul. 2013.

[7] M. Campista et al., "Routing metrics and protocols for wireless mesh networks," *IEEE Network*, vol. 22, no. 1, pp. 6–12, 2008.

[8] D. Pisinger, "The quadratic knapsack problem–a survey," *Discrete Applied Mathematics, Elsevier*, vol. 155, no. 5, pp. 623–648, Mar. 2007.

[9] R. Birke et al., "Architecture of a Network-Aware P2P-TV Application: The NAPA-WINE Approach," *IEEE Communications Magazine*, vol. 49, pp. 154–163, June 2011.

[10] S. Traverso et al., "Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems," in *12th IEEE Int. Conf. on Peer-to-Peer Computing (P2P-12)*, Tarragona, Spain, Sept. 2012, pp. 12–24.

[11] L. Abeni, C. Kiraly, and R. Lo Cigno, "Robust Scheduling of Video Streams in Network-Aware P2P Applications," in *IEEE ICC 2010*, Cape Town, ZA, May 2010, pp. 1–5.

[12] G. Gheorghe, A. Montresor, and R. Lo Cigno, "Security and Privacy Issues in P2P Streaming Systems: A Survey," *Springer Peer-to-Peer Networking and Applications*, vol. 4, pp. 75–91.

[13] G. Ciccarelli and R. Lo Cigno, "Collusion in Peer-to-Peer Systems," *Elsevier Comput. Netw.*, vol. 55, no. 15, pp. 3517–3532, Oct. 2011.

[14] T. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *IEEE INFOCOM 2002*, New York, NY, USA, June 2002, pp. 170–179.

[15] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, 2004, pp. 15–26.

[16] Y. Chen, X. Wang, C. Shi, E. K. Lua, X. Fu, B. Deng, and X. Li, "Phoenix: A weight-based network coordinate system using matrix factorization," *Network and Service Management, IEEE Trans. on*, vol. 8, no. 4, pp. 334–347, 2011.

[17] R. Fortuna et al., "QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs," in *10th IEEE Int. Conf. on Peer-to-Peer Computing (P2P-10)*, Delft, NL, Aug. 2010, pp. 1–10.

[18] A. Russo and R. Lo Cigno, "Delay-Aware Push/Pull Protocols for Live Video Streaming in P2P Systems," in *IEEE ICC 2010*, Cape Town, ZA, May 2010, pp. 1–5.

[19] S. Traverso et al., "Neighborhood Filtering Strategies for Overlay Construction in P2P-TV Systems: Design and Experimental Comparison," *IEEE/ACM Trans. on Networking*, vol. 23, no. 3, pp. 741–754, June 2015.

[20] S. Dolev, Y. Elovici, and R. Puzis, "Routing betweenness centrality," *J. ACM*, vol. 57, no. 4, pp. 25:1–25:27, May 2010.

[21] L. Maccari and R. Lo Cigno, "Waterwall: a cooperative, distributed firewall for wireless mesh networks," *EURASIP Jou. on Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–12.

[22] L. Maccari and R. Lo Cigno, "Betweenness estimation in OLSR-based multi-hop networks for distributed filtering," *Elsevier Jou. of Computer and System Sciences*, vol. 80, no. 3, pp. 670–685, May, 2014.

[23] A. Vzquez-Rodas and L. J. de la Cruz Llopis, "A centrality-based topology control protocol for wireless mesh networks," *Ad Hoc Networks*.

[24] Y. Sakata et al., "A Chunk Scheduling Based on Chunk Diffusion Ratio on P2P Live Streaming," in *IEEE NBiS 2012*, Sept. 2012, pp. 74–81.

[25] K.-L. Hua et al., "An efficient scheduling algorithm for scalable video streaming over P2P networks," *Elsevier, Computer Networks*, vol. 57, no. 14, pp. 2856–2868, Oct. 2013.

[26] J. Zhang et al., "Modeling and performance analysis of pull-based live streaming schemes in Peer-to-Peer network," *Elsevier Computer Communications*, vol. 40, pp. 22–32, Mar. 2014.

[27] L. Abeni, C. Kiraly, and R. Lo Cigno, "On the Optimal Scheduling of Streaming Applications in Unstructured Meshes," in *IFIP Networking*, Aachen, DE, May 2009.

[28] F. Harary, "Graph theory," 1969.

[29] H. D. Sherali and J. C. Smith, "An improved linearization strategy for zero-one quadratic programming problems," *Optimization Letters*, vol. 1, no. 1, pp. 33–47, 2007.

[30] U. Brandes and C. Pich, "Centrality estimation in large networks," 2007.

[31] R. Puzis, P. Zilberman, Y. Elovici, S. Dolev, and U. Brandes, "Heuristics for Speeding Up Betweenness Centrality Computation," in *Privacy, Security, Risk and Trust (PASSAT), 2012 Int. Conf. on and 2012 Int. Conf. on Social Computing (SocialCom)*, Sep. 2012, pp. 302–311.

[32] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Computer Aided Control Systems Design, 2004 IEEE Int. Symposium on*. IEEE, 2004, pp. 284–289.

[33] L. Maccari and R. Lo Cigno, "A week in the life of three large Wireless Community Networks," *Ad Hoc Networks, Elsevier*, vol. 24, Part B, pp. 175–190, Jan. 2015.

[34] L. Baldesi and L. Maccari, "NePA TesT: Network Protocol and Application Testing Toolchain for Community Networks," in *12th Conf. on Wireless On-demand Network Systems and Services (WONS)*, Cortina d'Ampezzo, IT, Jan. 2016.

Tube Streaming: Modelling Collaborative Media Streaming in Urban Railway Networks

Argyrios G. Tasiopoulos, Ioannis Psaras, Vasilis Sourlas and George Pavlou
Dept. of Electronic and Electrical Engineering, University College London, UK
Email: (argyrios.tasiopoulos, i.psaras, v.sourlas, g.pavlou)@ucl.ac.uk

***Abstract*—We propose a quality assessment framework for crowdsourced media streaming in urban railway networks. We assume that commuters either "tune in" to some TV/radio channel, or submit requests for content they desire to watch or listen to, which eventually forms a playlist of videos/podcasts/tunes. Given that connectivity is challenged by the movement of trains and the disconnection that this movement causes, users collaboratively download (through cellular and WiFi connections) and share content, in order to maintain undisrupted playback. We model collaborative media streaming for the case of the London Underground train network. The proposed quality assessment framework comprises a utility function which characterises the Quality of Experience (QoE) that users (subjectively) perceive and takes into account all the necessary parameters that affect smooth playback. The framework can be used to assess the media streaming quality in any railway network, after adjusting the related parameters.**

To the best of our knowledge, this is the first study to quantify the perceptual quality of collaborative media streaming in (underground) railway networks from a modelling perspective, as opposed to a systems perspective. Based on real commuter traces from the London Underground network, we evaluate whether audio and video can be streamed to commuters with acceptable QoE. Our results show that even with very high-speed Internet connection, users still experience disruptions, but a carefully designed collaborative mechanism can result in high levels of perceived QoE even in such disruptive scenarios.

***Index Terms*—Mobile Video/Audio Delivery, Collaborative Video/Audio Streaming, Crowdsourcing.**

I. Introduction

We envision ubiquitous streaming of popular channels, *e.g.,* national broadcasters, in smart city environments. We argue that there is a lot of space for resource optimisation in mixed cellular and WiFi access environments complemented with local transmission of content between mobile devices. Arguably, the quality of Internet services deteriorates when several hundreds of users attempt to connect simultaneously through the same WiFi Access Point (AP) or cellular Base Station (BS). This situation is rather common in large metropolitan areas where hundreds of users commute (*e.g.,* in trains or buses), wait in stations or airports, or just move slowly towards their destination [1]. The case becomes even more challenging when connectivity is physically disrupted, *e.g.,* when (underground) trains travel between stations. In those cases, installing more bandwidth will not necessarily improve performance [2], simply because quality often suffers due to: *i)* frequent handovers, *ii)* the hundreds of sessions that an AP/BS has to handle simultaneously, and *iii)* physical challenges, such as long disconnection periods. In urban railway networks, for instance, it is not uncommon the situation where in busy times more than 500 commuters are onboard a train (and might want to access online services), trains stay in stations (and therefore, connectivity is available) for as little as 20 seconds, followed by a disconnection period of 4 minutes (or more), when the train is in the tunnel. The situation is even worse for real-time streaming sessions, as opposed to downloading some static web-page content.

ISBN 978-3-901882-83-8

In the absence of any collaboration between users to download and stream content collectively, the users end up with degraded media quality and, more often than not, abandon their sessions.

Crowdsourcing-based approaches have been proposed recently in order to deal with the above challenges [3], [4], [5], [6]. Crowdsourced content retrieval is based on the premise that users share their storage, connectivity and energy resources given some direct or indirect incentives, *e.g.,* improved service quality [7], or some monetary reward [6], [8].

The implementation details of such a crowdsourced mobile video and audio on demand streaming service for commuters in urban railway networks have been studied in [3]. However, the question of *whether such a system can provide acceptable media experience (or QoE) to participating users has not been answered yet*. That said, in this study, we fill this gap by building a *quality assessment framework* that attempts to answer the above question.

Quality of Experience (QoE) is traditionally characterised by the subjective perception of users [9] and as such, it is difficult to quantify in terms of objective metrics [10]. Indeed, objective measures of user engagement, such as bitrate, throughput, startup delay and buffering, which by and large comprise the state of the art to date, are not adequate in a media-dominated Internet, let alone a mobile environment. The proposed framework, which is based on a utility function, extends our previous work in [5] and goes beyond the above primitive metrics to take into account the user's tolerance to disruptions, the energy needed to download and share media content, as well as the cost of using the cellular network. The energy factor, for instance, can influence user engagement in collaborative streaming in fear of battery depletion. In other words, users will not spend energy (and cellular data) to participate in a collaborative streaming system unless quality compensates. As such, we argue that a quality assessment framework for media delivery is necessary to characterise the

perceptual and subjective quality as opposed to the network-dependent, objective quality.

An urban railway system exhibits a number of challenges, as well as a number of physical properties, which can be utilised to make design decisions. For example, the train route and timetable is fixed (as opposed to walking users or vehicles). On the other hand, connectivity is physically disrupted when trains move between stations. Furthermore, disconnection periods can be as long as eight times this of the connection period, during which there is no WiFi connectivity and cellular connection is rather poor. Finally, the users' commuting patterns are unknown, but the aggregate number of commuters can be approximated depending on factors such as the time of day (*i.e.,* peak or off-peak time) and the direction of travel (*i.e.,* bound towards the city centre or not).

The main contributions of this work are:

- We define a utility function as the key aspect of our quality assessment framework. This utility function takes into account playback disruption and energy factors and provides a quantitative measure of a media delivery system for railway networks.
- We analyse a 17-day sample of commuters' journey traces, provided by Transport for London (TfL, `http://www.tfl.gov.uk`). We identify and illustrate commuting patterns and approximate the number of users travelling in each train throughout the day, as well as the connection and disconnection periods they experience.
- We quantify the users' experience by applying the utility function to the commuters' traces. Interestingly, we find that although it is difficult to achieve undisrupted multimedia playback, especially during the rush hours, a collaborative media streaming application can provide acceptable QoE.

The rest of the paper is organised as follows. In Section II, we discuss related works. Section III includes the analytical description of the problem under consideration, while in Section IV we give details of our commuter data trace and we evaluate the performance of the system. Finally, Section V concludes this study.

II. Related Work

The mobility properties of public transportation systems, as well as the limited connectivity during the commute makes communications in these environments a rather challenging problem. In the absence of any connectivity in underground train networks, authors in [11] build on the concept of *Familiar Strangers* [12] and analyse commuter traces to investigate whether collocation patterns exist. They find that such patterns indeed exist and build a content sharing and distribution network around content stored in commuters' devices. They identify users who posses content which is of interest to others and who will travel long enough within range of each other in order to successfully transfer the content in a *p2p* manner.

In [13] and [14] authors propose installation of a hub on public transportation vehicles, which commuters utilise in order to stay connected throughout their journeys. Although existence of a connectivity device on board the vehicle is desirable, the authors do not exploit aggregate connectivity and storage opportunities offered by user devices. Peer-to-peer communications between mobile devices have been extensively studied in the context of delay-tolerant networks, taking also advantage of social links to drive connectivity decisions [15], [16], but also with a target to improve QoE [7], [17].

None of these works, however, has attempted to model the performance of real-time multimedia streaming in challenged, intermittent connectivity environments and as such we consider these works complementary to ours.

Closer to our work are studies that investigate co-operative download techniques for mobile users. Such studies are applicable to a broader scope of problems in the context of delay-/disruption-tolerant networks. For instance, co-operative techniques that exploit cellular and local WiFi connectivity have been studied to improve the capacity available to mobile users, or in other words, increase the aggregated downlink rate of each user, *e.g.,* [6], [18], [19]. In [20], a group of collocated train commuters, using several wireless Internet access links, jointly access a video stream and each user contributes to the group by sharing his downloaded content. In [4], the authors formulate a network utility maximisation problem where a single static group of commuters attempts to access a video stream. According to [4], users try to utilise Internet access links as well as the device-to-device capacity by taking into account packet loss and applying network coding. The model in [4] is complementary to any study that investigates co-operative download for mobile devices. Finally, the authors in [3] design, implement, and evaluate *Microcast*, a system that improves the streaming experience of a group of users, albeit in relatively static conditions.

This second group of studies focuses mainly on *the implementation principles* of a co-operative streaming service, ignoring to a large extent the performance that such systems would achieve in reality. *Quality of Service* (QoS) metrics cannot be used for this purpose as they focus on network-related aspects and ignore the human perception factor. There is a clear lack in the literature of studies that characterise QoE for mobile media delivery. *Mean Opinion Score* (MOS) is the most widely used ranking metric that takes into account the objective opinion of individual users [9], [10]. In this study, we fill this gap by building a utility function that characterises the user's QoE when streaming media in train networks. Given the connectivity challenges in this case, tolerance to disruptions is a central factor. Furthermore, given that in order to participate in collaborative media streaming, users spend battery resources, energy consumption also plays a central role in the QoE assessment.

III. Quality Assessment Framework

A. System Model

We consider that users access the Internet through either WiFi or cellular links and can either individually *Pull*, or collectively *PUll and SHare* (PUSH) content. In this section we lay out the description of the model for each of the Internet access methods considered (*i.e.,* cellular and WiFi), as well as for each content retrieval approach (*i.e., Pull* or *PUSH*). We define as an epoch i, Ep^i, a time interval of duration $|Ep^i|$,

which consists of a connection period C^i (of duration $\lfloor C^i \rfloor$) and a disconnection, or poor connection quality period $\widetilde{C}^i$ (of duration $|\widetilde{C}^i|$). Clearly, $|Ep^i| = |C^i| + |\widetilde{C}^i|$, which also implies that a new epoch starts at each station, where the previous one finishes. Connection period is the time that a train spends in a station and disconnection period is the time it takes for the train to arrive at the next station.

In the simple case, users *Pull* content individually from the Internet. Video or audio content is split in chunks, where every chunk contains y seconds of playback time at bit-rate b. Users can also form groups to *PUll and SHare (PUSH)* content with fellow-commuters in the vicinity. Initially, we present the basic framework which effectively quantifies the utility obtained by individual users (Sections III-B, III-C). We then extend the utility function to incorporate aspects such as energy consumption and cellular download data charges (Section III-D). Finally, we adjust the basic framework to each of the content retrieval approaches (*i.e., Pull* and *PUSH*) and access method technologies in Sections III-E, III-F, III-G.

We assume that the functionality of the $PUSH$ approach, is realised in the context of a mobile Backend as a Service (mBaaS) platform that runs in the cloud (similar in rational to [3] and [6]) and is responsible for managing group synchronisation in terms of collaborative content retrieval. For the purposes of this study we ignore any potential implementation overhead (*e.g.,* chunk download scheduling). In particular, the mBaaS platform assigns to each member the content chunks that (s)he has to download and share with the rest of the group participants. All members of a group download and share equal number of chunks which implies *fairness* in terms of computation and communication effort. Our model notation is given on Table I.

$\lvert C^i\rvert$, $\lvert\widetilde{C}^i\rvert$	*i*-th connection/disconnection duration
$\lvert Ep^i\rvert$	*i*-th epoch duration
A	Internet access technology (WiFi,cellular)
R	Content retrieval approach (*Pull,PUSH, Hybrid*)
f	Epoch user started downloading the content
y, b, S	Playback time, bit-rate, and size of a chunk
Y	Content playback time duration
$X^i_{A,R}$	Chunks received at epoch *i* for A and R
$X^{f\to i}_{A,R}$	Total chunks received until epoch *i* (incl.)
$L^{f\to i}_{A,R}$	Total playback time received until epoch *i* (incl.)
$W^{f\to i}_{A,R}$	Playback time watched until epoch *i* (incl.)
$D^{f\to i}_{A,R}$	Playback Disruption until epoch *i* (incl.)
$U^{f\to i}_{A,R}$	User utility until epoch *i* (incl.)
$\widetilde{U}^{f\to i}_{A,R}$	User extended utility until epoch *i* (incl.)
$U^{C,f\to i}_{A,R}$, $U^{E,f\to i}_{A,R}$	User cellular and energy cost to epoch *i*
a_d,a_c,a_e	User disruption, cell, and energy sensitivity
X_{cell},X_{WiFi} X^g_{p2p}	Per second chunks delivery rate at cellular, WiFi, and sharing interface for a group g
$N(t)$	# of users requesting access at moment *t*
B_{WiFi}	Total bandwidth assigned to a platform of a station
B_{Cell}, $\hat{B}_{Cell}$	User good and poor cell bandwidth
$N_g(t)$	# of users of group g at moment *t*
B_{p2p}	Bandwidth limit for local *p2p* transfers
$V^{i,g}_{max(p2p)}$	Maximum amount of content to be shared and downloaded over an epoch i for group g
$V^{i,g}_A$, $\widetilde{V}^{i,g}_A$	Identical chunks downloaded by all the members of group g during good and poor connectivity

TABLE I: Model Notation

B. Playback and Playback Disruption

The Internet access medium (*i.e.,* WiFi or cellular), as well as the content retrieval approach (*i.e., Pull or PUSH*) affects the number of chunks that a user can receive over a predefined period of time. We denote as $X^i_{A,R}$ the number of chunks that a user receives over epoch i when (s)he accesses the Internet by technology A and retrieves a content by approach R.

Given reception of $X^i_{A,R}$ chunks, we calculate the *playback* and *playback disruption* periods that a user experiences over consecutive epochs. Firstly, we estimate the total number of chunks received by a user over epochs f to i as $X^{f\to i}_{A,R} = \sum_{j=f}^{i} X^j_{A,R}$, where f is the epoch during which the user enters the system.

Next, we express the number of chunks in terms of playback time. We assume that a chunk can be watched/listened only when it has been fully downloaded. Hence, the watching time worth of downloaded content from epoch f to epoch i for one user is:

$$L^{f\to i}_{A,R} = \lfloor X^{f\to i}_{A,R} \rfloor \times y, \tag{1}$$

where y is the playback duration of a content chunk.

Given that we model an on-demand streaming service, the downloaded playback time will differ from the actually watched playback time over epochs f to i, $W^{f\to i}_{A,R}$, due to buffered content. We model this difference in a retrospective manner between epochs i and $i-1$ and express the actually watched playback time $W^{f\to i}_{A,R}$ as the sum of: *i)* the watched playback time during the previous epochs $W^{f\to i-1}_{A,R}$ and *ii)* the difference between the total downloaded playback time between f and i, $L^{f\to i}_{A,R}$, and the actually watched playback time during the current epoch ($L^{f\to i}_{A,R} - W^{f\to i-1}_{A,R}$). Note that in case the user has buffered enough content to get through the current epoch, then $L^{f\to i}_{A,R} - W^{f\to i-1} = |Ep^i|$. Therefore, we have:

$$W^{f\to i}_{A,R} = \min[W^{f\to i-1}_{A,R} + \min(|Ep^i|, L^{f\to i}_{A,R} - W^{f\to i-1}_{A,R}), Y], \tag{2}$$

where Y is the total playback duration from the beginning of the journey, which apparently works as an upper bound of the watched time and $W^{f\to i-1}_{A,R} = 0$ for the first epoch.

Finally, the playback disruption time until epoch i, $D^{f\to i}_{A,R}$, is calculated as:

$$D^{f\to i}_{A,R} = \begin{cases} D^{f\to i-1}_{A,R}, & \text{if } W^{f\to i}_{A,R} = Y \\ \sum\limits_{j=f}^{i} |Ep^j| - W^{f\to i}_{A,R}, & \text{otherwise,} \end{cases} \tag{3}$$

where $D^{f\to i-1}_{A,R} = 0$ for the first epoch.

C. Utility Function

The playback time, $W^{f\to i}_{A,R}$, as well as the playback disruption, $D^{f\to i}_{A,R}$, are the fundamental components that we use in order to describe the user's utility in terms of QoE. Clearly, *the ideal utility of a user until epoch i is equal to the sum of the epochs' duration, or the content duration Y*, whichever is shorter:

$$U^{f\to i}_{ideal} = min(Y, \sum_{j=f}^{i} |Ep^j|). \tag{4}$$

We consider that the utility decreases due to the playback disruption $D_{A,R}^{f\to i}$ and express a *user's utility* in terms of undisrupted playback time according to the formula:

$$Utility\ Function:\ U_{A,R}^{f\to i} = W_{A,R}^{f\to i} - a_d \times D_{A,R}^{f\to i}, \quad (5)$$

where a_d is the user's tolerance to playback disruptions. The disruption tolerance factor decreases the user's utility by a_d. For instance, for a user who has watched 10 secs of undisrupted playback and therefore has built a utility function equal to 10, a disruption tolerance factor equal to 2 will decrease his utility to 8, after 1 second of disruption. We consider the disruption tolerance factor as a central component of the utility function for media delivery in connectivity-challenged mobile environments.

We express the "*efficiency*" of an Internet access technology A and content retrieval approach R for a commuter according to the following *utility ratio*:

$$Utility\ Ratio\ /\ Efficiency:\ Q_{A,R}^{f\to i} = \frac{T_d + U_{A,R}^{f\to i}}{T_d + U_{ideal}^{f\to i}}, \quad (6)$$

which is bounded by the interval $[0, 1]$. T_d is the "*Initial Tolerance Interval*", which indicates the user's patience to start-up delay. In the rest of this work, we use Eq. 6 to quantify the QoE that users obtain during their journeys, which together with the *utility function* in Eq. 5 comprise the two main building blocks of the proposed quality assessment framework.

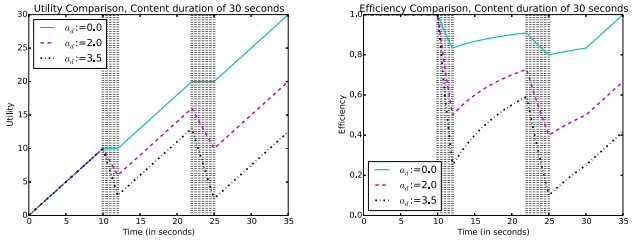
Fig. 1 illustrates the *utility* (Eq. 5) and *efficiency* (Eq. 6) fluctuation for 3 users with disruption tolerance (*i.e.,* a_d) 0, 2, and 3.5, respectively; the users experience a playback disruption of 2 seconds after 10 seconds of playback time and another disruption of 3 seconds after playback time of 20 seconds; the total playback duration is 30 seconds, which also means that the ideal utility is equal to 30. Apparently, delay sensitivity equal to 0 leads to efficiency equal to 1 (Fig. 1b), while for delay sensitivity equal to 2 and 3.5 in this setting the produced efficiency is 0.666 and 0.416, respectively. This result demonstrates that *efficiency* is subjective when it comes to QoE, since it is subject to the users' temporal utility/satisfaction and their personal tolerance to disruptions.

The *Utility Ratio or Efficiency* can get negative values due to extended disruptions. In this case we assume that a rational user would quit the attempt to watch (listen) this video (music playlist), which would set the Efficiency's value to zero, or more formally:

$$Q_{A,R}^{f\to i,t} = max(\frac{T_d + U_{A,R}^{f\to i,t}}{T_d + U_{ideal}^{f\to i,t}}, 0). \quad (7)$$

D. Cellular and Energy Cost

To further extend our model with a realistic representation of a working system we incorporate two more factors in the utility function of users. The first relates to the medium used to download content and is associated with the cost of using the cellular network, denoted as *cellular cost*. The second factor relates to the energy consumed in order to use this medium, denoted as *energy cost*. Given the description and structure of the utility function in Eq. 5 the cellular and energy costs need to be converted to actual playback time. That said, the cellular cost is the equivalent of the playback time downloaded through the cellular network, denoted as $U_{A,R}^{C,f\to i}$ for epochs f to i.



(a) Utility Comparison (b) Efficiency Comparison

Fig. 1: Comparison of user utilities and efficiencies over time for a disruption of 2 and 3 seconds after 10 and 20 seconds of playback time.

As regards the energy cost, an interface supporting radio technologies works in different *power states*, each one related to a different workload as well as power consumption. These states include the *Idle* state, when no Internet connection is required, and consumes the least possible energy. Starting from this state, an interface can be promoted to one or more productive states by spending a fixed amount of energy and time, where productive means that an interface becomes able to receive or transmit data. Finally, an interface experiences a tail power phenomenon where it stays in a high power state, in anticipation of more data exchange, before returning to its initial *Idle* state.

We denote as $E_{A,R}^{prom}$ and $E_{A,R}^{tail}$ the fixed energy cost of all involved interfaces, of technology A and approach R, when being promoted to a productive state and when experiencing the tailing phenomenon, respectively. Thus, the energy spent until epoch i, $E_{A,R}^{f\to i}$, is:

$$E_{A,R}^{f\to i} = E_{A,R}^{prom} + E_{A,R}^{prod,i} + E_{A,R}^{tail} + E_{A,R}^{f\to i-1}, \quad (8)$$

where $E_{A,R}^{prod,i}$ is the energy spent on the productive state of all interfaces for receiving and sharing content during epoch i. Therefore, by identifying the technology, A^*, and approach, R^*, which spend the smallest possible amount of energy until epoch i, $E_{A^*,R^*}^{f\to i}$, we can express the energy cost of a candidate technology, A, and approach, R, in terms of playback time, $U_{A,R}^{E,f\to i}$, by:

$$U_{A,R}^{E,f\to i} = (\frac{E_{A,R}^{f\to i}}{E_{A^*,R^*}^{f\to i}} - 1) \times L_{A^*,R^*}^{f\to i}. \quad (9)$$

The value of $U_{A,R}^{E,f\to i}$ in Eq. 9 is effectively the additional content (in terms of playback time) that a user would download if (s)he used technology, A^*, and approach, R^*.

Building on Eq. 5, the extended utility function that integrates the energy and cellular costs is:

$$\widetilde{U}_{A,R}^{f\to i} = U_{A,R}^{f\to i} - a_c \times U_{A,R}^{C,f\to i} - a_e \times U_{A,R}^{E,f\to i}, \quad (10)$$

where a_c and a_e are the corresponding weights of cell and energy cost, called *cell* and *energy sensitivity*.

In the following, we first discuss the general chunk reception rates over WiFi and cellular connectivity, which are independent of the content retrieval approach (Section III-E). Then we adjust those reception rates to the *Pull* and *PUSH* cases, in

Sections III-F and III-G, respectively. The target is to define the total chunks received until epoch i, that is, the $\lfloor X_{A,R}^{f \to i} \rfloor$ component of Eq. 1, which then feeds Eq. 2 and eventually the Utility Function defined in Eq. 5 and extended in Eq. 10.

E. Chunk Reception Rate over WiFi and Cellular

The size of a single chunk, S, is $S = b \times y$, where b is the bitrate and y is the chunk's playback duration.[1] Then, assuming interference-free cellular downlinks, the cellular chunk delivery rate (per second) will be $X_{cell}(t) = B_{cell}/S$, where B_{cell} is the bandwidth allocated to the user by the cellular network provider. For simplicity, we consider that the cellular bandwidth is stable, irrespectively of the number of active users.[2] We assume that B_{cell} is the bandwidth availability when the train is stopped at some station and that the connection quality deteriorates when the train is moving (*i.e.,* $\ddot{B}_{cell} < B_{cell}$).

Without loss of generality, we assume that the available WiFi access bandwidth per platform of each station is equal to B_{WiFi}, which is equally shared among the users/commuters at each platform. Therefore, the chunk reception rate per second and per user at time t, received through the WiFi AP of a platform is, $X_{WiFi}(t) = \frac{B_{WiFi}}{N(t) \cdot S}$, where $N(t)$ is the number of users requesting Internet access at moment $t \in [t_i, t_{i+1})$ at the given platform. Please note that there is no WiFi connectivity during disconnection periods (*i.e.*, when trains are in-between stations), which means that the chunk reception rate is: $\ddot{X}_{WiFi} = 0$.

In the *PUSH* approach, apart from the chunks received through the Internet, users receive chunks from group members too. The chunk reception rate between group members in the *PUSH* approach is proportional to the group size. Chunks are shared over bandwidth B_{p2p} and the number of chunks that can be shared among a group g of $N_g(t)$ members at a moment t, $X_{p2p}^g(t)$, is defined as:

$$X_{p2p}^g(t) = \frac{B_{p2p}}{N(t) \cdot S} \times N_g(t), \tag{11}$$

Eq. 11 implies an underlying pseudo-broadcast sharing mechanism (similar to [3]) where a single peer at a time unicasts its content to another peer, while the rest of the members of the group overhear the transmission. Note that the mBaaS platform (mentioned earlier in Section III-A) is also responsible for organising the "sharing turns", given that only one user can unicast at a time.

F. Pull Reception Rate

In the simple *Pull* approach users *pull* content individually. For cellular Internet access, the total number of chunks received during epoch i is equal to:

$$X_{cell,Pull}^i = |C^i| \times X_{cell} + |\widetilde{C}^i| \times \ddot{X}_{cell}. \tag{12}$$

In the WiFi AP case, where the medium is shared between users, the total number of chunks that a user receives through a WiFi AP over epoch i is:

$$X_{WiFi,Pull}^i = \int_{t=t_i}^{t_i+|C^i|} X_{WiFi}(t)dt, \tag{13}$$

where t_i is the starting time of epoch i.

Finally, in case users utilise both interfaces for Internet access, in a *Hybrid* way, the total number of chunks they receive during epoch i is:

$$X_{Hybrid,Pull}^i = X_{cell,Pull}^i + X_{WiFi,Pull}^i. \tag{14}$$

G. PUll and SHare (PUSH) Reception Rate

In the *PUSH* approach, users who belong to a group g of $N_g(t)$ members, at moment t, share their downloaded content. The chunk reception rate from fellow group members is given in Eq. 11. Our model does not include multi-hop transmissions which implies that all members of a group have to be within transmission range of each other. For that purpose, we consider WiFi Direct as the technology of choice in order to transmit in long distances with high rates [21]. We also assume that devices can make simultaneous use of two separate half-duplex WiFi interfaces, one for downloading through the WiFi AP and another one for local sharing. Our approach is also applicable in the simple scenario where only one WiFi interface is available per device, but in that case downloading and sharing should take place sequentially, that is, the users would first download the required chunks and then share them with the rest of the group.

The total volume of content that can be shared between a group (according to Eq. 11) is X_{p2p}^g, or more formally, *the theoretical maximum amount of content that can be shared over an epoch i for a group g, $V_{max(p2p)}^{i,g}$, is:*

$$V_{max(p2p)}^{i,g} = \int_{t=t_i}^{t_i+|Ep^i|} X_{p2p}^g(t)dt. \tag{15}$$

On the other hand, *the maximum volume of content that can be downloaded collaboratively over the connection period of an epoch by access approach A, $V_{A,C}^{i,g}$, is:*

$$V_{A,C}^{i,g} = \int_{t=t_i}^{t_i+|C^i|} X_A(t) \times N_g(t)dt, \tag{16}$$

where $X_A(t)$ is the number of downloaded chunks per second per group member, during the connection period.

Finally, the corresponding maximum downloaded content under poor connectivity (*i.e.,* using cellular connection when the train is in-between stations), $V_{A,\widetilde{C}}^{i,g}$, is:

$$V_{A,\widetilde{C}}^{i,g} = \int_{t=t_i+|C^i|}^{t_i+|Ep^i|} \ddot{X}_A(t) \times N_g(t)dt, \tag{17}$$

where $\ddot{X}_A(t)$ is the number of downloaded chunks per second per group member, during poor connection (or disconnection)

[1] Note that we do not consider dynamic rate adaptation (*e.g.*, DASH) for simplicity of modelling. We evaluate the proposed model under the lowest possible rate, hence, any higher bandwidth availability will only increase the performance we observe.

[2] In reality, even the cellular bandwidth assigned to each user can be influenced by the level of contention (that is, number of users), but this happens for larger number of users, possibly in the order of thousands, which is not the case of a train (station).

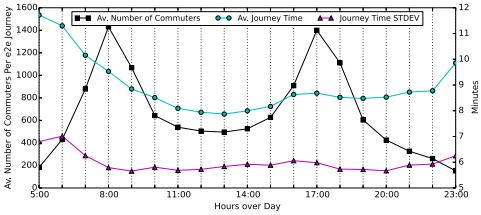


Fig. 2: Average journey duration and number of commuters per end-to-end journey of the line.

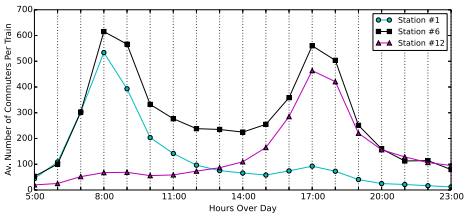


Fig. 3: Number of commuters per train (single direction) for a station in the beginning, middle, and end of the line.

period. It follows that *the total volume of content that can be downloaded over an entire epoch i, $X^i_{A,PUSH}$, is:*

$$X^i_{A,PUSH} = V^{i,g}_{A,C} + V^{i,g}_{A,\widetilde{C}}. \quad (18)$$

Eq. 18 effectively expresses the total number of chunks that the group can download *and* share within epoch i. In case $X^i_{A,PUSH} > V^{i,g}_{max(p2p)}$ the group has downloaded exactly as much content as it can share during epoch i. This would happen when the Internet access bandwidth is higher than the *p2p* one and therefore, users can download faster than they can share. In this case, users continue downloading individually without sharing.

IV. Evaluation

A. Commuter Journey Traces Dataset

We analyse the anonymised commuter trace dataset from London's Oyster Radio Frequency Identification cards. The dataset is a 17-day trace of all journeys made in all of the 11 lines of the London Underground network; the total number of journeys in our dataset is in the order of one million per day. Each journey is identified by an entry and exit point in the network, which is recorded at the granularity of one minute. The traces do not include information regarding the specific lines used by commuters, hence, in order to get an insight of how many passengers are on board a train at some given point throughout the day, we build a custom simulator which takes into account both the entry/exit points of commuters, but also the specific topology of the network. That is, we also consider the distance between stations, as well as the intersections of lines at each station and the routes that each line follows.

Given that a journey's route might cross more than one lines, and that commuters follow the shortest path with the fewest interchanges between their entry and exit points, we process the trace and assign commuters to specific lines and then to individual trains.

Due to space limitations and for better visualisation of our results, we choose one line and present results for this line only. However, we report that evaluations with most of the lines of the network present similar results.

B. Group Formation Insights/Potential

Fig. 2 depicts the average number of commuters that our chosen line serves per one end-to-end journey. Obviously, not all commuters travel from one end to the other, but this plot includes all commuters that at some point in their journey use this line of the network. The number of commuters is averaged over all trains of the day in per hour time slots. We observe that there are two clear peaks during the morning and the afternoon rush hours. During those times, one train can serve up to 1400 commuters in its end-to-end journey. Depending on their entry and exit points, these commuters can form groups to *PUll and SHare* content from the network.

In order to get a closer view of the potential to form groups, in Fig. 2, we also depict the average journey time per commuter, as well as the standard deviation of the per commuter journey time. According to this plot, commuters use this line on average for approximately 9 minutes with a standard deviation of around 5 minutes. Although the proportion of time that commuters physically share journeys depends on how much the chosen routes overlap (*i.e.,* entry and exit points), this time is enough to watch or listen to the headline stories of some TV/radio channel.

Clearly, from Fig. 2 we see that a line serves different volumes of commuters at different times of day, but their distribution over the stations of the end-to-end line needs further investigation. A more detailed analysis though, shows that commuters' volumes differ hugely at each station depending on the train direction and the time of day. We note that the lines normally start and terminate at the outskirts of the city centre, but always cross the city centre itself. In Fig. 3 we present the average number of commuters per train at three different stations over the duration of a day. The three stations chosen are towards the beginning of the line (Station 1), the middle of the line (a central location, Station 6) and the end of the line (Station 12). Interestingly, and somewhat expectedly, we see that the station at the beginning of the line serves most users in the morning rush hours (*i.e.,* users move towards the city centre), whereas the station towards the other end of the line serves most users during the afternoon rush hours (*i.e.,* on the way back). The station in the city centre presents two peaks one in the morning and one in the afternoon rush hour.

This result makes clear the need for provisioning of Internet streaming services according to journey patterns, station locations and the time of day. That is, given that connectivity is available only when trains are within stations, where they normally stay for about 20-30 seconds and that disconnection periods last 1-4 minutes, the amount of content that needs to be buffered in order to avoid playback disruptions differs hugely. Given also that different stations are busy at different times of day depending on the area and the direction of the train, the system has to be carefully modelled to include these factors in order to guarantee undisrupted playback.

Standard Setting Variable	Value
Total bandwidth per station, B_{WiFi}	0.5 Gbps
Cell rate - moving ($\ddot{B}_{cell}$)/static (B_{cell}) train	150-550 Kbps
Sharing bandwidth, B_{p2p}	54 Mbps
Connection period duration for each station i, $\|C^i\|$	20"
Participation probability, p_{list}	50%
Music/Video bit-rate	160/419 Kbps
Chunk playback time, y	5"
Playlist duration, Y	15'
Zipf's exponent, a	1
Playlists generated per station, Z	5-95
Delay (a_d), Energy (a_e), Cell (a_c) Sensitivity	3, 0, 0

TABLE II: Standard Evaluation Setting

C. Simulation and Evaluation Setup

We assume that users have access to both WiFi connectivity when trains are in stations and cellular connectivity throughout their journeys. Although in the case of the London Underground cellular connectivity is not available, here we include this access option for completeness. We also assume that users can use both their cellular and their WiFi interfaces simultaneously to receive content (*Hybrid* approach).

Although the current WiFi access deployment at the London Underground network provides download speeds of up to 100Mbps, here, we consider a future, overprovisioned network where each station is connected to the Internet with 500Mbps links. This bandwidth is split between all platforms and among all lines passing through each station.

For the cellular case, we assume that users get between 150Kbps and 550Kbps - closer to the lower value when the train is moving and to the highest one when the train is static in the station. Finally, we set the users' sensitivity to delay (a_d in Eq. 5) equal to 3, which means that for every second of disruption the users' utility decreases by three, whereas their utility increases by one for every second of undisrupted playback. This value is specifically chosen as an extreme scenario, where users are not tolerant to disruptions.

In addition to the different access methods, we evaluate the users' QoE (*i.e.,* Eq. 6) when two different types of content are available, that is, music content (at 160Kbps) and video content (at 419Kbps). The chosen bit rates are the lowest possible for acceptable quality streaming. In cases where the system can achieve minimum disruptions, an adaptive increase of the corresponding bit rates can be applied (*e.g.,* through DASH), but this is out of the scope of this paper. Unless mentioned otherwise, the full list of settings of our evaluation setup is given on Table II.

Due to the limitations of the environment under investigation, we do not assume that users individually choose to stream content and only if requests match, then users form groups. This would clearly result in very few and very small groups, effectively reflecting the *Pull* case. Instead, we evaluate two specific scenarios. In the first one, we assume that users create music or video "playlists" according to genre preferences, *e.g.,* sports clips, or jazz music. Users then join a group and add their own preferences to the list. We assume that between 5 and 95 playlists are generated in each epoch, resulting in more than 500 playlists available throughout the train's end-to-end journey. The playlists' duration is set to 15 minutes to reflect the average journey time plus standard deviation. In our second scenario, users tune in to radio or TV channels [22], [23]

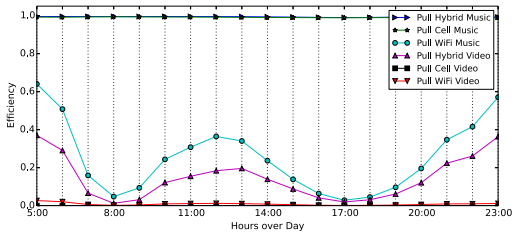


Fig. 4: 15-min Playlists scenario *Pull* efficiency for WiFi, Cell, and Hybrid access methods.

and we assume that 50 channels are supported by the system, which would cover the main broadcasters of a country. The difference between the *playlist* and *channel* scenario is that in case of the latter, users form larger groups. In both cases, users choose which playlist or channel to subscribe to following a Zipf distribution [24].

In both scenarios, and after the users have created new playlists at each station, these playlists are suggested to the rest of the commuters who are not participating in any other group. A commuter is interested in this list and participates with a probability p_{list} while choosing the r-th element of the list according to a Zipf distribution with probability $f(r; a, Z)$ where a is the Zipf exponent and Z is the number of elements in the list.

We measure the users' perceptual QoE as expressed through Eq. 6. The initial tolerance interval (T_d in Eq. 6) is set to 5 seconds, after the expiration of which each user abandons the attempt to receive content. This setting is based on studies that assess the patience of users to load web content (*e.g.*, [25]).

D. Performance Evaluation

1) Pull Approach - Playlists scenario: Fig. 4 illustrates the average (over all trains) efficiency of *WiFi*, *Cell*, and *Hybrid* access methods of the *Pull* approach for music and video streaming over one operational day of the tube network. Given that with the *Cell* approach each user gets between 150Kbps and 550Kbps and that the bitrate for streaming music rate is 160Kbps, it is straightforward that the network can support such a service. Hence, the *Cell* approach (and stemming from that, obviously, the *Hybrid* approach) can provide good QoE. On the other hand, the *WiFi* access method experiences disruptions which increase as the volume of commuters increases (*i.e.,* during rush hours - see Fig. 2 and Fig. 3). In the case of video streaming all three access approaches, *WiFi*, *Cell* and *Hybrid* perform worse than music streaming. In the following we exclude the music bitrate for the cell access approach, as it clearly can be supported when cell access is available (not for the case of the London Underground network though).

2) PUSH Approach - Playlists scenario: From Fig. 5 we observe that streaming music over the WiFi network is challenged by disruptions/disconnections, achieving performance close to 0.8 at best, while during rush hours this can go as low as 0.4. In the case of Video playlists, which is even more demanding (*i.e.,* higher bit rates), achieves lower performance ranging between 0.7-0.8 when utilising both the WiFi and the cellular connection. Note that the performance

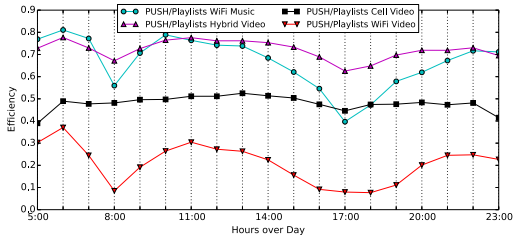


Fig. 5: 15-min Playlists scenario *PUSH* efficiency for WiFi, Cell, and Hybrid access methods.

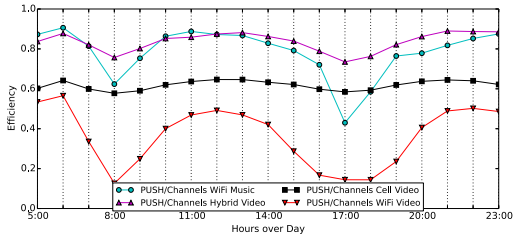


Fig. 6: Streaming Channels - *PUSH* efficiency for WiFi, Cell, and Hybrid access methods.

of the *PUSH* approach is acceptable during off-peak time, especially considering the extreme delay sensitivity assumed here. Overall, based on the parameters used here (especially for the delay sensitivity) collaborative download can provide acceptable service quality, especially when it is combined with cellular connectivity. In the following we experiment with different parameters to investigate whether quality can improve further.

3) PUSH Approach - Channels scenario: Efficiency performance follows similar trends in the case of the second scenario (*i.e.*, TV/radio channel streaming - see Fig. 6), where, however, we observe a small increase in the QoE perceived by the users. This is because, in the "channel" case, groups form for longer time periods [26] and also more users join in as trains move towards their destination. That said, groups are bigger in size compared to the playlists scenario, giving the opportunity to move more content locally. Of course, during the rush hours the available WiFi bandwidth per commuter is significantly small and despite the size of the formed groups the efficiency is still quite low, since each user can download and share a very small portion of the desired content. As we show next, it requires a large amount of available WiFi bandwidth at each station in order to increase the performance of the system. Furthermore, another important factor that affects performance is the initial tolerance interval as we show later in this section.

4) PUSH Approach - WiFi Bandwidth Factor: In general more bandwidth available at each platform/station improves performance, but here we examine whether investing more on bandwidth would pay off in terms of users' QoE. In Fig. 7 we present the case for video streaming to groups of commuters (*i.e.*, *PUSH* approach) for the WiFi and the Hybrid access methods.

Interestingly, the efficiency achieved by the WiFi access method during peak times overtakes the one achieved during off-peak times when the available bandwidth at each station is significantly large ($\geq$ 1Gbps). In those cases, larger number

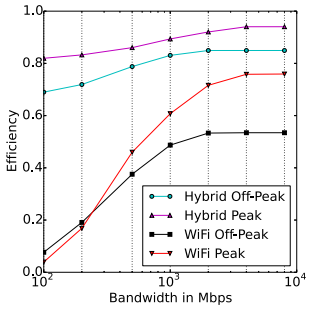


Fig. 7: 15-min Video Playlists scenarion *PUSH* Efficiency for WiFi and Hybrid access methods for increasing bandwidth (Peak time: 8am, Off-Peak: 11am).

of commuters on board the trains result in larger groups. In turn, each group gets a larger stake of the available bandwidth. Overall, we see that even when 1Gbps is readily available at each of the hundreds of tube stations throughout the network, it is still difficult to achieve uninterrupted video streaming relying on the WiFi connectivity only, even at off-peak times. On the other hand, when collaborative streaming is combined with cellular connectivity, we observe performance close to 0.8, even for smaller bandwidth values and rather disruption-sensitive users (*i.e.*, $a_d = 3$).

5) PUSH Approach - Cell and Energy Sensitivity: The cell sensitivity integrates the cost factor, that is, the monetary cost to download through the cellular network (*i.e.*, a_c in Eq. 10), as opposed to the WiFi access. In general the efficiency decreases linearly in all the retrieval approaches (*i.e., Pull* and *PUSH*) with respect to the cell sensitivity. In more details in the $Pull$ case it declines with exactly the same rate during both peak and off-peak times, since the amount of data that each commuter downloads from the cellular interface in each case remains the same. On the other hand, for the $PUSH$ approach (Fig. 8) the efficiency decline rate is less steep and the difference between the peak and off-peak time is proportional to the average group size. In this approach users also exchange a significant amount of data minimizing the data to be directly downloaded from the cellular network.

We use the power state machine presented in [27] to evaluate the energy sensitivity of the cellular and WiFi interfaces of smartphone devices for each one of the power state (*i.e.,* Promotion, Productive and Tail). Our findings show that despite the fact that *PUSH* uses an additional interface for sharing data, the low promotion and tail energy required by the sharing interface decreases the overall performance as we increase the energy sensitivity. Finally, we notice that increasing the sharing energy transmission coefficient, a_{tr}, causes only slight performance decline, proportional to the coefficient's actual value (Fig. 9). This is partly because local transfers (through the sharing interface) completes much faster, therefore, spending little time in "transmission mode".

6) PUSH Approach - Tolerance Interval: The *"Initial Tolerance Interval"* of users indicates the patience of users to startup delay. Throughout our evaluation this interval was set to 5 seconds, according to related studies on users' tolerance to delays [25]. In this experiment, we investigate the effect of

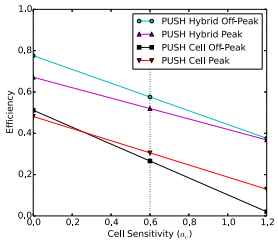


Fig. 8: Cell Sensitivity Increase - Scenario 1: Video Playlist

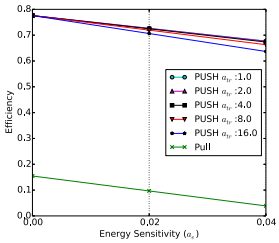


Fig. 9: Energy Sensitivity Increase - Scenario 1 Video Playlist, Hybrid

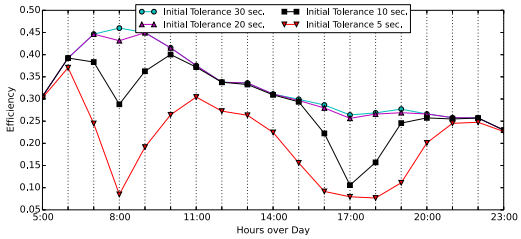


Fig. 10: 15-min Playlists scenario PUSH- WiFi Efficiency for different Initial Tolerance Intervals.

the tolerance interval on users' QoE (in Fig. 10); we assume a 15-min playlist when users are pulling and sharing content over WiFi and the tolerance interval is set to 10, 20 and 30 seconds. As the users' tolerance increases, we observe that the QoE increases too. This is a straightforward result, given that the groups formed in this case are larger and can therefore, get larger share of the available WiFi bandwidth. However, we also observe that after the 20-second threshold the performance does not improve further.

V. Conclusions

We have designed a model that characterises the QoE of users in an urban railway network, when they attempt to stream real-time media content. Our model applies to both simple *Pull* cases, and to collaborative download, *PUll and SHare (PUSH)* cases and assumes WiFi connectivity in train stations, as well as cellular connection throughout the journey. The QoE is expressed in terms of the efficiency that users enjoy and takes into account connection and disconnection periods, sensitivity to disruptions and to the energy spent to download content, as well as the cost to use the cellular network. We have analysed commuters' traces and have applied our model to these mobility patterns. We found that it is difficult to maintain undisrupted playback, especially in case of high bit rates, *i.e.*, video content, but at the same time, well thought-out collaborative mechanisms can increase the perceived QoE even under such challenged conditions. When cellular connectivity is available, performance improves considerably, given that users can utilise both interfaces (the WiFi and the cellular one) simultaneously.

Acknowledgments

V. Sourlas is supported by the EC through the FP7-PEOPLE-IEF INTENT project, (GA no. 628360). The rest of the authors are supported by the EU-Japan initiative under EC FP7 GreenICN project (GA no. 608518, NICT no. 167), the EC H2020 UMOBILE project (GA no. 645124) and the EPSRC INSP Fellowship (no. EP/M003787/1).

References

[1] G. Lyons and K. Chatterjee, "A human perspective on the daily commute: Costs, benefits and trade-offs," *Transport Reviews*, 2008.
[2] L. Li, K. Xu, D. Wang, C. Peng, Q. Xiao, and R. Mijumbi, "A measurement study on tcp behaviors in HSPA+ networks on high-speed rails," in *IEEE INFOCOM 2015*.
[3] L. Keller and et al., "Microcast: Cooperative video streaming on smartphones," in *MobiSys '12*.
[4] H. Seferoglu, L. Keller, B. Cici, A. Le, and A. Markopoulou, "Cooperative video streaming on smartphones," in *Allerton Conference*, 2011.
[5] A. G. Tasiopoulos, I. Psaras, and G. Pavlou, "Mind the gap: modelling video delivery under expected periods of disconnection," in *ACM CHANTS '14*.
[6] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, A. Korakis, and T. Leandros, "Bits and coins: Supporting collaborative consumption of mobile internet," in *IEEE INFOCOM '15*.
[7] C.-L. Tsao and R. Sivakumar, "On effectively exploiting multiple wireless interfaces in mobile hosts," in *ACM CoNEXT '09*.
[8] W. Wu, R. Ma, and J. Lui, "Distributed caching via rewarding: An incentive scheme design in P2P-VoD systems," *Parallel and Distributed Systems, IEEE Transactions on*, 2014.
[9] M. Venkataraman and M. Chatterjee, "Inferring video qoe in real time," *Network, IEEE*, 2011.
[10] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A quest for an internet video quality-of-experience metric," in *ACM HotNets-XI*, 2012.
[11] L. McNamara, C. Mascolo, and L. Capra, "Media sharing based on colocation prediction in urban transport," in *ACM MobiCom '08*.
[12] E. Paulos and E. Goodman, "The familiar stranger: anxiety, comfort, and play in public places," in *ACM SIGCHI '04*.
[13] J. LeBrun and C.-N. Chuah, "Bluetooth content distribution stations on public transit," in *Workshop on Decentralized resource sharing in mobile computing and networking*, 2006.
[14] F. P. Tso and et al., "Dragonnet: A robust mobile internet service system for long-distance trains," *Mobile Computing, IEEE Transactions on*, 2013.
[15] C. Boldrini, M. Conti, and A. Passarella, "Exploiting users' social relations to forward data in opportunistic networks: The HiBOp solution," *Pervasive and Mobile Computing*, 2008.
[16] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, 2011.
[17] G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath, "Combine: leveraging the power of wireless peers through collaborative downloading," in *ACM MobiSys '07*.
[18] S. Ioannidis, A. Chaintreau, and L. Massoulié, "Optimal and scalable distribution of content updates over a mobile social network," in *IEEE INFOCOM 2009*.
[19] J. Whitbeck, M. Amorim, Y. Lopez, J. Leguay, and V. Conan, "Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays," in *IEEE WoWMoM '11*.
[20] M. Stiemerling and S. Kiesel, "A system for peer-to-peer video streaming in resource constrained mobile environments," in *ACM U-NET Workshop '09*.
[21] "WiFi alliance. Wi-Fi direct: http://www.wi-fi.org."
[22] "PPLive: http://www.pplive.com."
[23] "TVUnetworks: http://www.tvunetworks.com."
[24] "Cisco. cisco visual networking index: Global mobile data traffic forecast update, 2013-2018. white paper, [online] http://goo.gl/l77haj, 2014."
[25] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, "Web site delays: How tolerant are users?" *Journal of the Association for Information Systems*, 2004.
[26] F. Wang, J. Liu, and Y. Xiong, "Stable peers: Existence, importance, and application in peer-to-peer live video streaming," in *IEEE INFOCOM '08*.
[27] N. Ding, D. Wagner, X. Chen, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *ACM SIGMETRICS '13*.

Energy-Aware Cooperative Computation in Mobile Devices

Ajita Singh, Yuxuan Xing, Hulya Seferoglu
ECE Department, University of Illinois at Chicago
asingh64@uic.edu, yxing7@uic.edu, hulya@uic.edu

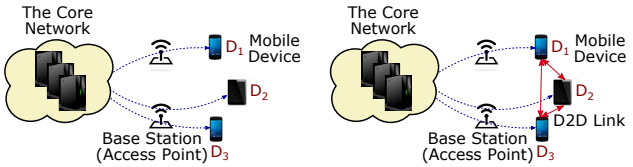
***Abstract*—New data intensive applications, which are continuously emerging in daily routines of mobile devices, significantly increase the demand for data, and pose a challenge for current wireless networks due to scarce resources. Although bandwidth is traditionally considered as the primary scarce resource in wireless networks, the developments in communication theory shifts the focus from bandwidth to other scarce resources including processing power and energy. Especially, in device-to-device networks, where data rates are increasing rapidly, processing power and energy are becoming the primary bottlenecks of the network. Thus, it is crucial to develop new networking mechanisms by taking into account the processing power and energy as bottlenecks. In this paper, we develop an energy-aware cooperative computation framework for mobile devices. In this setup, a group of cooperative mobile devices, within proximity of each other, (i) use their cellular or Wi-Fi (802.11) links as their primary networking interfaces, and (ii) exploit their device-to-device connections (*e.g.,* Wi-Fi Direct) to overcome processing power and energy bottlenecks. We evaluate our energy-aware cooperative computation framework on a testbed consisting of smartphones and tablets, and we show that it brings significant performance benefits.**

I. Introduction

The dramatic increase in mobile applications and the number of devices demanding for wireless connectivity poses a challenge in today's wireless networks [1], [2], and calls for new networking mechanisms.

One of the promising solutions to address the increasing data and connectivity demand is Device-to-Device (D2D) networking. As illustrated in Fig. 1(a), the default operation in current wireless networks is to connect each device to the Internet via its cellular or Wi-Fi interface. The D2D connectivity idea, which is illustrated in Fig. 1(b), breaks this assumption: it advocates that two or more devices in close proximity can be directly connected, *i.e.,* without traversing through auxiliary devices such as a base station or access point. D2D networking, that can be formed by exploiting D2D connections such as Wi-Fi Direct [3], is a promising solution to the ever increasing number and diversity of applications and devices. In this context, it is crucial to identify scarce resources and effectively utilize them to fully exploit the potential of D2D networking.

Although bandwidth is traditionally considered as the primary scarce resource in wireless networks, in D2D networks, thanks to close proximity among devices and the developments in communication theory, the main bottleneck shifts from bandwidth to other scarce resources including processing power and energy. Next, we present our pilot study demonstrating that processing power can be more pronounced as a bottleneck than bandwidth in D2D networks.



(a) The default operation (b) D2D connectivity

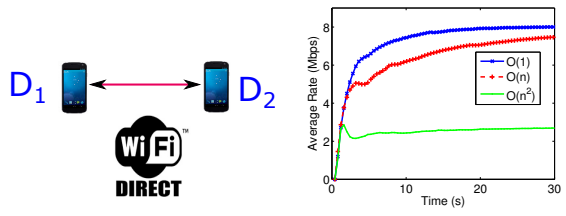
Fig. 1. (a) The default operation for the Internet connection. (b) D2D connectivity: two or more mobile devices can be connected directly, *i.e.,* without traversing through the core network, if they are in close proximity by exploiting local area connections such as Wi-Fi Direct.

Pilot Study: We developed a prototype for this pilot study as shown in Fig. 2(a), where a mobile device D_2 receives data from another device D_1 over a Wi-Fi Direct link. We use Android operating system [4] based Nexus 7 tablets [5] as mobile devices. In this experiment, after receiving the packets, the mobile device D_2 performs operations with complexities of $\mathcal{O}(1)$, $\mathcal{O}(n)$, and $\mathcal{O}(n^2)$ above the transport layer (TCP), where n is the packet size, and the operations we perform are counting the bytes in the packets. In particular, $\mathcal{O}(1)$, $\mathcal{O}(n)$, and $\mathcal{O}(n^2)$ correspond to (i) no counting, (ii) counting every byte in a packet once, and (iii) counting every byte in a packet n times, respectively. We demonstrate in Fig. 2(c) the received rate at the mobile device D_2 (note that this is the rate we measure at the mobile device D_2 after performing computations) versus time. This figure demonstrates that the received rate decreases significantly when the complexity increases. □

Our pilot study shows that even if actual bandwidth is high and not a bottleneck, processing power could become a bottleneck in D2D networks. Similar observations can be made for the energy bottleneck as detailed in our technical report [6]. Furthermore, with the advances in communication theory, *e.g.,* millimeter wave communication [7], it is expected that data rates among devices in close proximity will increase significantly, which will make processing power and energy more pronounced as bottlenecks. However, existing applications, algorithms, and protocols are mainly designed by assuming that bandwidth is the main bottleneck. Thus, it is crucial to develop new networking mechanisms when bandwidth is not the primary bottleneck, but processing power and energy are.

Thus, in this paper, our goal is to create group of devices that help each other cooperatively by exploiting high rate D2D

ISBN 978-3-901882-83-8 © 2016 IFIP



(a) Setup (b) Rate vs Time

Fig. 2. **Pilot Study:** (a) Setup: Data is transmitted from mobile device D_1 to another mobile device D_2. In this setup, the mobile devices are Android operating system (OS) [4] based Nexus 7 tablets [5]. The specific version of the Anroid OS is Android Lollipop 5.1.1. The devices have 16GB storage, 2GB RAM, Qualcomm Snapdragon S4 Pro, 1.5GHz CPU, and Adreno 320, 400MHz GPU. Packet size is $500B$. (b) Transmission rate versus time for different computational complexities at the receiver side. Note that we present the rate that we measure at the mobile device after performing the computations. The presented rates are the averages over 10 seeds.

connections to overcome the processing power and energy bottlenecks. The next example demonstrates our approach.

Example 1: Let us consider Fig. 1(a) again, where device D_1 would like to receive a file from a remote resource via its cellular or Wi-Fi connection. Assume that the cellular (or Wi-Fi) rates of all devices are 1Mbps, but device D_1 can receive data with 500kbps rate due to processing power bottleneck, *i.e.,* device D_1 has limited processing power (similar to our pilot study we presented earlier). In a traditional system, D_1 will behave as a single end point, so its receiving rate will be limited to 500kbps. On the other hand, if devices D_1, D_2, and D_3 will behave as a group and cooperate, then devices D_2 and D_3 can also receive and process 500kbps portions of data, and transmit the processed data to device D_1 over D2D connections. This increases the receiving rate of device D_1 to 1.5Mbps from 500kbps, which is a significant improvement.

This example could be extended for scenarios when energy (battery of mobile devices) is limited. For example, if device D_2's battery level is too low, its participation to the group activity should be limited. □

Application Areas. The scenario in the above motivating example could arise in different practical applications from health, education, entertainment, and transportation systems. The following are some example applications. *Health:* A person may own a number of health monitoring devices (activity monitoring, hearth monitoring, etc.) which may need updates from the core network. These updates - potentially coded for error correction, compression, and security reasons - should be processed (decoded) by these devices. Processing takes time, which may lead to late reaction to the update (which may require timely response) and energy consumption. On the other hand, by grouping mobile devices, the person's smartphone or tablet could receive the update, process, and pass the processed data to the health monitoring devices via high rate D2D links. *Education & Entertainment:* A group of students may want to watch the video of a lecture from an online education system (or an entertainment video) while sitting together and using several mobile devices. In this setup, one of the devices can download a base layer of a video and decode, while the other devices could download enhancement layers and decode. The decoded video layers could be exchanged among these mobile devices via high rate D2D links. As in the motivating example, if one device's download and decoding rate is limited to 500kbps, it could be improved to 1.5Mbps with the help of other devices.

Note that the processing overhead in these applications could be due to any computationally intensive task related to data transmission. For example, for video transmission applications, H.264/AVC decoders introduce higher computational complexity when higher quality guarantees are needed [8], [9]. Another example could be network coding; for example, data could be network coded at the source to improve throughput, error correction, packet randomization potential of network coding [10]. However, most of the network coding schemes introduce high computational complexity at the receiver side; $\mathcal{O}(n^3)$, [11], [12], which limits the transmission rate. Encryption could be another example that introduces processing overhead [13].

Thus, there exist several applications and scenarios where bandwidth and energy could be bottlenecks, while bandwidth is not the bottleneck. This makes our approach demonstrated in Example 1 crucial. In particular, in this paper, we develop an *energy-aware cooperative computation* framework for mobile devices. In this setup, a group of cooperative mobile devices, within proximity of each other, (i) use their cellular or Wi-Fi (802.11) links as their primary networking interfaces, and (ii) exploit their D2D connections (Wi-Fi Direct) for cooperative computation. Our approach is grounded on a network utility maximization (NUM) formulation of the problem and its solution [14]. The solution decomposes into several parts with an intuitive interpretation, such as flow control, computation control, energy control, and cooperation & scheduling. Based on the structure of the decomposed solution, we develop a stochastic algorithm; *energy-aware cooperative computation*.[1] The following are the key contributions of this work:

- We consider a group of cooperative mobile devices within proximity of each other. In this scenario, we first investigate the impact of processing power to transmission rate. Then, we develop an energy-aware cooperative computation model, where devices depending on their energy constraints could cooperate to get benefit of aggregate processing power in a group of cooperative devices.
- We characterize our problem in a NUM framework by taking into account processing power, energy, and bandwidth constraints. We solve the NUM problem, and use the solution to develop our stochastic algorithm; *energy-aware cooperative computation (EaCC)*. We show that EaCC provides stability and optimality guarantees.
- An integral part of our work is to understand the performance of EaCC in practice. Towards this goal, we

[1]Note that our work focuses on cooperative resource utilization in mobile devices. In this sense, our work is complementary to and synergistic with: (i) creating incentive mechanisms in D2D networks, and (ii) providing privacy and security for D2D users [15], [16]. Looking into the future, it is very likely that our proposed work on the design, analysis, and implementation of cooperative resource utilization is gracefully combined with the work on creating incentives and providing privacy and security.

develop a testbed consisting of Nexus 5 smartphones and Nexus 7 tablets. All devices uses Android 5.1.1 as their operation systems. We implement *EaCC* in this testbed, and evaluate it. The experimental results show that our algorithm brings significant performance benefits.

The structure of the rest of the paper is as follows. Section II presents related work. Section III gives an overview of the system model. Section IV presents the NUM formulation of our cooperative computation scheme. Section V presents our stochastic algorithm; *EaCC*. Section VI evaluates the performance of our scheme in a real testbed. Section VII concludes the paper.

II. Related Work

This work combines ideas from D2D networking, network utility maximization, and stochastic network control.

The idea of D2D networking is very promising to efficiently utilize resources, so it has found several applications in the literature. In particular, D2D connections are often used to form cooperative groups for data streaming applications, and for the purpose of (i) content dissemination among mobile devices [17], [18], (ii) cooperative video streaming over mobile devices [19], [20], [21], [22], and (iii) creating multiple paths and providing better connectivity by using multiple interfaces simultaneously [23], [24]. As compared to this line of work, we investigate the impact of processing power and energy in D2D networks, and develop mechanisms to effectively utilize these scarce resources.

D2D networking is often used for the purpose of offloading cellular networks. For example, previous work [25], [26], [17] disseminate the content to mobile devices by taking advantage of D2D connections to relive the load on cellular networks. Instead of offloading cellular networks, our goal is to create energy-aware cooperation framework to overcome the processing power and energy bottlenecks of mobile devices.

There is an increasing interest in computing using mobile devices by exploiting connectivity among mobile devices [27]. This approach suggests that if devices in close proximity are capable of processing tasks cooperatively, then these devices could be used together to process a task as it is a cheaper alternative to remote clouds. This approach, sparking a lot of interest, led to some very interesting work in the area [28], [29], [30]. As compared to this line of work, we focus on processing power and energy bottlenecks in mobile devices and address the problem by developing energy-aware cooperative computation mechanism.

An integral part of our proposed work in this task is to develop efficient resource allocation mechanisms. In that sense, our approach is similar to the line of work emerged after the pioneering work in [31], [32], [33]. However, our focus is on energy-aware cooperative computation.

III. System Model

We consider a cooperative system setup with N mobile devices, where $\mathcal{N}$ is the set of the mobile devices. Our system model for three nodes are illustrated in Fig. 3(a). The source in Fig. 3(a) represents the core network and base stations (access points). This kind of abstraction helps us focus on the bottlenecks of the system; processing power, energy of mobile devices, and downlink/uplink data rates. In this setup, mobile devices communicate via D2D connections such as Wi-Fi Direct, while the source communicates with mobile devices via cellular or Wi-Fi links. We consider in our analysis that time is slotted and t refers to the beginning of slot t.

Connecting Devices Together: The total flow rate towards device n in Fig. 3(a) (as also explained in Fig. 3(b)) is $\sum_{k\in\mathcal{N}} x_{n,k}(t)$, where $x_{n,n}(t)$ is the transmission rate of the packets from the source towards device n, and these packets will be used by device n. Note that $x_{n,k}(t)$ is the transmission rate of the packets from the source towards device n, and these packets will be processed by device n and forwarded to device k. On the other hand, $y_n(t)$ is the total flow rates targeting device n as demonstrated in Fig. 3(b). The source constructs a queue $S_n(t)$ for the packets that will be transmitted to the mobile device n. The evolution of $S_n(t)$ based on $y_n(t)$ and $x_{k,n}(t)$ is expressed as

$$S_n(t+1) \leq \max[S_n(t) - \sum_{k\in\mathcal{N}} x_{k,n}(t), 0] + y_n(t), \qquad (1)$$

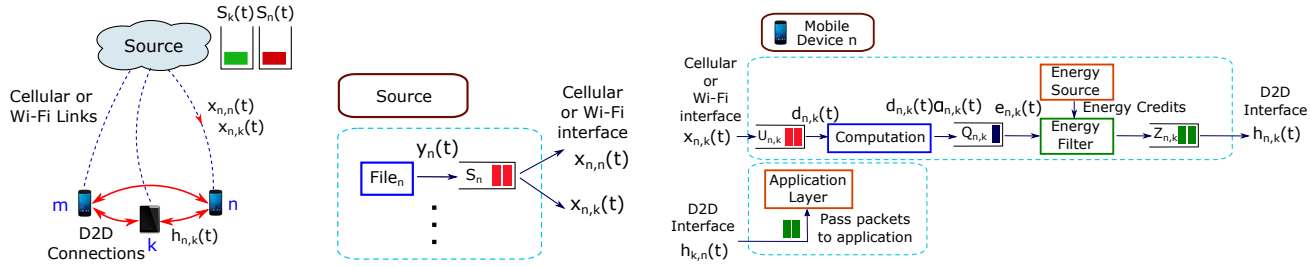
where the inequality comes from the fact that there may be less than $y_n(t)$ packets arriving into $S_n(t)$ at time t in practice (*e.g.,* in real time applications, the number of available packets for transmission could be limited).

The flow rate $y_n(t)$ is coupled with a utility function $g_n(y_n(t))$, which we assume to be strictly concave function of $y_n(t)$. This requirement is necessary to ensure stability and utility optimality of our algorithms. The ultimate goal in our resource allocation problem is to determine the flow rates; $y_n(t)$ which maximize the sum utility $\sum_{n\in\mathcal{N}} g_n(y_n(t))$.

Finally, flow rate over D2D connection between device n and k is $h_{n,k}(t)$, $k \neq n$. Note that $h_{n,k}(t)$ is to help node k using node n as a processing device.

Inside a Mobile Device: In each device, we develop different modules depending on where data is arriving from (as shown in Fig. 3(c)); *i.e.,* from the source via cellular or Wi-Fi interface, or other mobile devices via D2D interfaces.

When data is arriving from a D2D interface, it is directly passed to the application layer, as this data is already processed by another device. On the other hand, when data is arriving from the source via cellular or Wi-Fi interfaces, packets go through multiple queues as shown in Fig. 3(c), where $U_{n,k}$, $Q_{n,k}$, and $Z_{n,k}$ represent three different queues constructed at mobile device n for the purpose of helping node k. Incoming packets via cellular or Wi-Fi links are stored in $U_{n,k}$, which then forwards the packets to *computation* block with rate $d_{n,k}(t)$. The computation block processes the packets, and pass them to queue $Q_{n,k}$. Note that the output rate from computation block is $d_{n,k}(t)\alpha_{n,k}(t)$, where $\alpha_{n,k}(t)$ is a positive real value. This value captures any possible rate changes at the computation block, *i.e.,* $\alpha_{n,k}(t)$ is a rate shaper. For example, if the computation block is H.264/AVC decoder or transcoder, we expect that the rate at the output of the computation block should be higher than the input. Thus, $\alpha_{n,k}(t)$ captures this fact for any n, k, t. On the other hand, if there is no rate change



(a) System Model (b) Building blocks of the source. (c) Building blocks of mobile device n

Fig. 3. (a) System model for the scenario of three devices; n, m, k. The source in this model represents the core network and base stations (access points). (b) Building blocks of the source. File$_n$, $\forall n$ is read and inserted in the buffer $S_n(t)$, and packets are transmitted from $S_n(t)$. $x_{n,k}(t)$ is the transmission rate of the packets from the source towards device n, and these packets will be processed by device n and forwarded to device k. (c) Building blocks of mobile device n. If packets are received from the source via cellular and Wi-Fi interfaces, then they go to the computation and energy control blocks. If packets are received from other mobile devices via D2D interface, they are directly passed to the application.

after the processing, then $\alpha_{n,k}(t) = 1$.

The processed (and possibly rate shaped) packets are queued at $Q_{n,k}(t)$ and passed to *energy filter*. The energy filter is coupled to the energy source, which determines the amount of energy that can be spent to support the tasks at each slot. The amount of energy is determined according to *energy credits*. In particular, the energy source, depending on the battery level as well as the estimate on the expected battery consumption in the near future, calculates the number of packets that can be supported by the mobile device, and the same number of energy credits enter the energy filter. (Note that both energy filter, energy source, and energy credits are not real, but virtual entities, so they can be modeled by using a few counters in practice.) Thus, at each transmission slot, packets are transmitted from $Q_{n,k}(t)$ to $Z_{n,k}(t)$ with rate $e_{n,k}(t)$ if there exist energy credits in the energy filter. Finally, packets from $Z_{n,k}(t)$ are transmitted to application if device n is the destination of the data (*i.e.,* $n = k$), or they are transmitted to the original destination via D2D interface with rate $h_{n,k}(t)$.

The computation and energy filter blocks in Fig. 3(c) model the processing and energy bottlenecks of the mobile device, respectively. If packets in $U_{n,k}$ increases too much, this means that the computation block, hence processing power, is the bottleneck, so node n should not receive much packets from the source. Similarly, if $Q_{n,k}$ increases too much, this means that energy filter is the bottleneck, so again node n should not receive much packets. Note that there could be also some buildup in $Z_{n,k}$ if the link between node n and k is the bottleneck of the system, and it should be taken into account when the energy-aware cooperative computation framework is developed.

Also, it is crucial in our system model to put energy filter after the computation block, because if device n will help device k, the actual amount of packets that are supposed to be transmitted are the processed packets, which will cause energy consumption (*i.e.,* not the packets before processing).

Based on the above intuitions and observations, we will develop our resource allocation problem and algorithm in the next sections. The evolution of the queues $U_{n,k}(t)$, $Q_{n,k}(t)$, and $Z_{n,k}(t)$ are provided in Table I.

Links: In our system model, we consider two scenarios: (i) cellular + Wi-Fi Direct, and (ii) Wi-Fi + Wi-Fi Direct.

TABLE I
EVOLUTION OF QUEUES $U_{n,k}(t)$, $Q_{n,k}(t)$, AND $Z_{n,k}(t)$.

$U_{n,k}(t+1) \leq \max[U_{n,k}(t) - d_{n,k}(t), 0] + x_{n,k}(t)$
$Q_{n,k}(t+1) \leq \max[Q_{n,k}(t) - e_{n,k}(t), 0] + d_{n,k}(t)\alpha_{n,k}(t)$
$Z_{n,k}(t+1) \leq \max[Z_{n,k}(t) - h_{n,k}(t), 0] + e_{n,k}(t)$

In both cases, the D2D links between mobile devices are Wi-Direct. In the first case, *i.e.,* in cellular + Wi-Fi Direct, the links between the source and mobile devices are cellular, while they are Wi-Fi in the second case, *i.e.,* in Wi-Fi + Wi-Fi Direct. These two scenarios are different from each other, because in the first scenario, cellular and Wi-Fi Direct links could operate simultaneously as they use different parts of the spectrum. On the other hand, in the second scenario, both Wi-Fi and Wi-Fi Direct use the same spectrum, so they time share the available resources. Our model and energy-aware cooperative computation framework are designed to operate in both scenarios. Next, we provide details about our link models.[2]

In the system model in Fig. 3(a), each mobile device $n \in \mathcal{N}$ is connected to the Internet via its cellular or Wi-Fi link. At slot t, $\boldsymbol{C}^s(t)$ is the channel state vector of these links, where $\boldsymbol{C}^s(t) = \{C_1^s(t), ..., C_n^s(t), ..., C_N^s(t)\}$. We assume that $C_n^s(t)$ is the state of the link between the source and mobile device n, and it takes "ON" and "OFF" values depending on the state of the channel. Without loss of generality, if mobile device n does not have Internet connection, then $C_n^s(t)$ is always at "OFF" state, which means there is no cellular or Wi-Fi connection.

Since we consider that mobile devices are in close proximity and transmission range, they form a fully connected clique topology. At slot t, $\boldsymbol{C}^w(t)$ is the channel state vector of the D2D links, where $\boldsymbol{C}^w(t) = \{C_{1,2}^w(t), ..., C_{n,k}^w(t), ..., C_{N-1,N}^w(t)\}$. We assume that $C_{n,k}^w(t)$ is the state of the D2D link between node n and k.

We consider protocol model in our formulations [34], where each mobile device can either transmit or receive at the same time at the same frequency. Assuming that $\boldsymbol{C}(t) =$

[2]Note that the link models described in this section provide a guideline in our algorithm development and basis in our theoretical analysis. However, in Section VI, we relax the link model assumptions we made in this section, and evaluate our algorithms on real devices and using real links.

$\{\boldsymbol{C}^s(t), \boldsymbol{C}^w(t)\}$ is the channel state vector of the system including both the links between the source and mobile devices as well as among mobile devices, $\Gamma_{\boldsymbol{C}(t)}$ denotes the set of the link transmission rates feasible at time slot t depending on our protocol model. In particular, for cellular + Wi-Fi Direct setup, $\Gamma_{\boldsymbol{C}(t)}$ is the set that allows more links to operate at the same time, while for the Wi-Fi + Wi-Fi Direct setup, $\Gamma_{\boldsymbol{C}(t)}$ is a more limited set due to the interference among the links.

IV. Problem Formulation

In this section, we characterize the stability region of the energy-aware cooperative computation problem, and formulate network utility maximization (NUM) framework. The solution of the NUM framework provides us insights for developing the stochastic control algorithms in the next section.[3]

A. Stability Region

We provide the stability region of the cooperative computation system for both cellular + Wi-Fi Direct and W-Fi + Wi-Fi Direct setups. First, the flow conservation constraint at the source should be $y_n \leq \sum_{k\in\mathcal{N}} x_{k,n}$ to stabilize the system. This constraint requires that the total outgoing rate from the source, *i.e.,* $\sum_{k\in\mathcal{N}} x_{k,n}$ should be larger than the generated rate y_n.

Furthermore, the following flow conservation constraints inside a mobile device should be satisfied for stability; $x_{n,k} \leq d_{n,k}$, $d_{n,k}\alpha_{n,k} \leq e_{n,k}$, and $e_{n,k} \leq h_{n,k}$. These constraints are necessary for the stability of queues $U_{n,k}$, $Q_{n,k}$, and $Z_{n,k}$, respectively. Finally, the transmission rates over the links should be feasible, *i.e.,* $\{x_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N}, k\in\mathcal{N}} \in \Gamma_{\boldsymbol{C}}$.

Thus, we define the stability region as $\Lambda = \{\{y_n, x_{n,k}, d_{n,k}, e_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N},k\in\mathcal{N}} \mid y_n,\ x_{n,k},\ d_{n,k},\ e_{n,k},\ h_{n,k} \geq 0,\ \forall n \in \mathcal{N}, k \in \mathcal{N},\ y_n \leq \sum_{k\in\mathcal{N}} x_{k,n},\ x_{n,k} \leq d_{n,k},\ d_{n,k}\alpha_{n,k} \leq e_{n,k},\ e_{n,k} \leq h_{n,k},\ \{x_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N},k\in\mathcal{N}} \in \Gamma_{\boldsymbol{C}}\}$.

B. NUM Formulation

Now, we characterize our NUM problem.

$$\max_{\boldsymbol{y}} \sum_{n\in\mathcal{N}} g_n(y_n)$$
$$\text{s.t. } y_n, x_{n,k}, d_{n,k}, e_{n,k}, h_{n,k} \in \Lambda,\ \forall n \in \mathcal{N}, k \in \mathcal{N} \quad (2)$$

The objective of the NUM problem in (2) is to determine $y_n, x_{n,k}, d_{n,k}, e_{n,k}, h_{n,k}$ for $\forall n \in \mathcal{N}, k \in \mathcal{N}$ which maximize the total utility $\sum_{n\in\mathcal{N}} g_n(y_n)$.

C. NUM Solution

Lagrangian relaxation of the flow conservation constraints that characterize the stability region Λ gives the following Lagrange function:

$$L = \sum_{n\in\mathcal{N}} g_n(y_n) - \sum_{n\in\mathcal{N}} s_n(y_n - \sum_{k\in\mathcal{N}} x_{k,n}) - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} u_{n,k}(x_{n,k} - d_{n,k}) - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} q_{n,k}(d_{n,k}\alpha_{n,k} - e_{n,k}) - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} z_{n,k}(e_{n,k} - h_{n,k}) \quad (3)$$

where s_n, $u_{n,k}$, $q_{n,k}$, and $z_{n,k}$ are the Lagrange multipliers. Note that we will convert these Lagrange multipliers to queues S_n, $U_{n,k}$, $Q_{n,k}$, and $Z_{n,k}$ when we design our stochastic algorithm in the next section.

The Lagrange function in (3) is decomposed into sub-problems such as flow, computation, and energy controls as well as cooperation and scheduling. The solutions of (3) for y_n, $d_{n,k}$, $e_{n,k}$, $x_{n,k}$, and $h_{n,k}$ are expressed as:

$$\underline{\text{Flow control:}}\ \max_{\boldsymbol{y}} \sum_{n\in\mathcal{N}} (g_n(y_n) - y_n s_n) \quad (4)$$

$$\underline{\text{Computation control:}}\ \max_{\boldsymbol{d}} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} d_{n,k}(u_{n,k} - q_{n,k}\alpha_{n,k}) \quad (5)$$

$$\underline{\text{Energy control:}}\ \max_{\boldsymbol{e}} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} e_{n,k}(q_{n,k} - z_{n,k}) \quad (6)$$

$$\underline{\text{Cooperation \& Scheduling:}}$$
$$\max_{\boldsymbol{x},h} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} [x_{n,k}(s_k - u_{n,k}) + z_{n,k}h_{n,k}]$$
$$\text{s.t. } \{x_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N},k\in\mathcal{N}} \in \Gamma_{\boldsymbol{C}} \quad (7)$$

Next, we design a stochastic algorithm; energy-aware cooperative computation inspired by the NUM solutions in (4), (5), (6), (7).

V. Energy-Aware Cooperative Computation

Now, we provide our energy-aware cooperative computation algorithm which includes *flow control*, *computation control*, *energy control*, and *cooperation & scheduling*.

Energy-Aware Cooperative Computation (EaCC):

- *Flow Control:* At every time slot t, $y_n(t)$ is determined by maximizing $\max_{\boldsymbol{y}} [Mg_n(y_n(t)) - S_n(t)y_n(t)]$ subject to $y_n(t) \leq R_n^{\max}$, where $R_n^{\max}$ is a positive constant larger than the transmission rate from the source, and M is a large positive constant. Note that $S_n(t)$ is the queue size at the source of flow and stores packets that are supposed to be transmitted to mobile device n. After $y_n(t)$ is determined, $y_n(t)$ packets are inserted in queue $S_n(t)$ (as illustrated in Fig. 3(a)).
- *Computation Control:* At every time slot t, the computation control algorithm at device n determines $d_{n,k}(t)$ by optimizing

$$\max_{\boldsymbol{d}} \sum_{k\in\mathcal{N}} d_{n,k}(t)[U_{n,k}(t) - Q_{n,k}(t)\alpha_{n,k}(t)]$$
$$\text{s.t. } \sum_{k\in\mathcal{N}} d_{n,k}(t) \leq D_n^{\max} \quad (8)$$

[3]Note that NUM optimizes the average values of the parameters that are defined in Section III. By abuse of notation, we use a variable, *e.g.,* ϕ as the average value of $\phi(t)$ in our NUM formulation if both ϕ and $\phi(t)$ refers to the same parameter.

where $D_n^{\max}$ is a positive constant larger than the processing rate of the computation block in device n dedicated to help device k. The interpretation of (8) is that at every time slot t, $d_{n,k^*} = D_n^{\max}$ packets are passed to the computation block (in Fig. 3(b)) if $U_{n,k^*}(t) - Q_{n,k^*}(t) > 0$, where k^* is the mobile device that maximizes (8). Otherwise, no packets are sent to the computation block. The packets that are being processed by the computation block are passed to $Q_{n,k}(t)$. Note that some computation blocks may require to receive a group of packets to be able to process them. In that case, $D_n^{\max}$ is arranged accordingly (*i.e.,* it can be increased to transfer a group of packets).

- *Energy Control:* At every time slot t, the energy control algorithm at device n determines $e_{n,k}(t)$ by optimizing

$$\max_{\boldsymbol{e}} \sum_{k\in\mathcal{N}} e_{n,k}(t)[Q_{n,k}(t) - Z_{n,k}(t)]$$
$$\text{s.t.} \sum_{k\in\mathcal{N}} e_{n,k}(t) \leq E_{n,k}^{\max} \quad (9)$$

  where $E_n^{\max}$ is a positive constant larger than the energy capacity of device n dedicated to help device k. The interpretation of (9) is that at every time slot t, $e_{n,k^*} = E_n^{\max}$ packets are passed to the energy filter (as illustrated in Fig. 3(b)) if $Q_{n,k^*}(t) - Z_{n,k^*}(t) > 0$, where k^* is the mobile device that maximizes (9). Otherwise, no packets are sent to the energy filter. The packets passing through the energy filter are inserted in $Z_{n,k}(t)$.
- *Scheduling & Cooperation:* At every time slot t, the scheduling and cooperation algorithm determines transmission rates over links, *i.e.,* $x_{n,k}(t)$ and $h_{n,k}(t)$ by maximizing

$$\max_{\boldsymbol{x},\boldsymbol{h}} \sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{N}} [x_{n,k}(t)(S_k(t) - U_{n,k}(t)) + h_{n,k}(t)Z_{n,k}(t)]$$
$$\text{s.t. } \boldsymbol{x}, \boldsymbol{h} \in \Gamma_{\boldsymbol{C}(t)} \quad (10)$$

  For cellular + Wi-Fi Direct system, (10) is decomposed into two terms: maximizing $\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{N}} x_{n,k}(t)(S_k(t) - U_{n,k}(t))$ and $\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{N}} h_{n,k}(t)Z_{n,k}(t)$, because cellular and Wi-Fi Direct transmissions operate simultaneously and transmission over one link does not affect the other. On the other hand, for Wi-Fi + Wi-Fi Direct setup, the joint optimization in (10) should be solved.
  Note that transmissions over all links are unicast transmissions in our work, where unicast is dominantly used in practice over cellular, Wi-Fi, and Wi-Fi Direct links. Also, it is straightforward to extend our framework for broadcast transmissions.

Theorem 1: If channel states are i.i.d. over time slots, and the arrival rates $E[y_n(t)] = A_n, \forall n \in \mathcal{N}$ are interior of the stability region Λ, then energy-aware cooperative computation stabilizes the network and the total average queue sizes are bounded.

Furthermore, if the channel states are i.i.d. over time slots, and the traffic arrival rates are controlled by the flow control algorithm of energy-aware cooperative computation, then the admitted flow rates converge to the utility optimal operating point with increasing M.

Proof: The proof is provided in [6]. ■

Our energy-aware cooperative computation framework has several advantages: (i) distributed, (ii) takes into account scarce resources such as processing power and energy in addition to bandwidth to make control decisions, and (iii) utilizes available resources; processing power, energy, and bandwidth in a utility optimal manner. Theorem 1 shows the theoretical performance guarantees of our framework, while we focus on its performance in a practical setup in the next section.

VI. Performance Evaluation

In this section, we evaluate our *energy-aware cooperative computation* (EaCC) scheme using a testbed that consists of Android based smartphones and tablets. The evaluation results show that our scheme significantly improves throughput as compared to (i) no-cooperation, where each device receives its content from the source without cooperating with other devices, and (ii) cooperation, where multiple mobile devices cooperate, but the cooperating devices do not do computation and energy control for other devices (mobile devices just receive packets from the source, and relay them to other mobile devices without processing and energy control). Next, we present testbed setup and results in detail.

A. Setup & Implementation Details

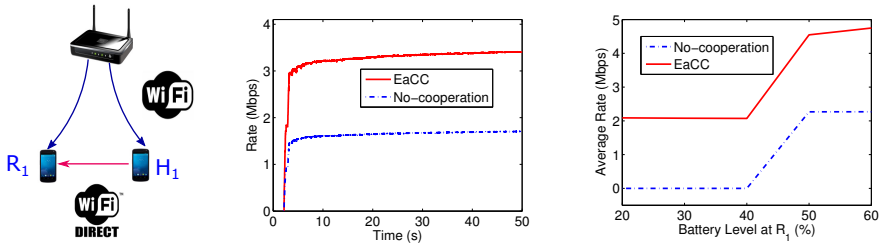
Devices: We implemented a testbed of the setup shown in Fig. 3(a) using real mobile devices, specifically Android 5.1.1 based Nexus 5 smartphones and Nexus 7 tablets.

We classify devices as (i) a source device, which acts as the source in Fig. 3(a), (ii) helper devices, which receive data from the source, process it, and transmit to other devices (receivers) to help them, and (iii) receiver devices, which receive data from both the source device and the helpers. A receiver device processes data arriving from the source, but it does not process the data arriving from helpers as the helpers send already processed data.[4] Note that a device could be both receiver and a helper device depending on the configuration.

Integration to the Protocol Stack: We implemented our energy-aware cooperative computation (EaCC) framework as a slim layer between transport and application layers. In other words, we implemented our framework on top of TCP. This kind of implementation has benefits, because (i) mobile devices do not need rooting, and (ii) our framework and codes could be easily transferred to mobile devices using other operating systems such as iOS.

Source Configuration and EaCC Implementation: We implemented the source node in Fig. 3 using a Nexus 5 smartphone. Basically, multiple files; File$_n$, File$_k$ requested by devices n and k are read by using the public java class *BufferedInputStream* according to the flow control algorithm described in Section V and shown in Fig. 3(b). The bytestream

[4]Note that we relax this assumption in our technical report [6] for multimedia applications.



(a) System Model (b) Rate vs. Time (c) Rate vs. Energy

Fig. 4. (a) System model consisting of a source device, one receiver, and one helper. Wi-Fi is used between the source and the receiver device (R_1) and the helper device (H_1), while Wi-Fi Direct is used to connect R_1 to H_1. (b) Average rate versus time for the setup shown in (a) for the case that all the devices are Android based Nexus 7 tablets. (c) Average rate versus energy level at receiver device R_1. In this setup, all devices are Android based Nexus 5 smartphones. In both (b) and (c), the average rate is calculated as the average over 10 trials (with different seeds). The computation under consideration in this experiment is $O(n^2)$, which counts the number of bytes in a packet for each byte in the packet (*i.e.,* recursive counting).

is packetized by setting each packet to $500B$, and packets are inserted into source buffers; $S_n(t)$, $S_k(t)$. We set the flow control parameters as; $M = 500$, $R_n^{\max} = 100$, and slot duration is $20msec$. We used log function as our utility function. In this setup, reading files, converting bytestream into packets, and inserting packets into the input queues are done by multiple threads, *i.e.,* a thread runs for each file; File$_n$ in Fig. 3(b).

The other set of threads at the source device make packet transmission decisions from the source device to receiver and helper devices. In particular, the source node collects $U_{n,k}(t)$ information from all mobile devices. At each time slot, the source node checks $S_k(t)-U_{n,k}(t)$, and if $S_k(t)-U_{n,k}(t) > 0$, then 100 packets are transmitted from $S_k(t)$ to the TCP socket at the source device for transmission to mobile device n.

EaCC Operation on Mobile Devices: All mobile devices (including helper or receiver+helper devices) implement all the building blocks illustrated Fig. 3(c). Multiple threads are used to make these blocks operating simultaneously.

The first thread at mobile device n receives packets that are transmitted by the source node, and inserts these packets in $U_{n,k}$.

The second thread has two tasks. First, it transfers packets from $U_{n,k}$ to $Q_{n,k}$ according to the computation control algorithm in (8), where $D_n^{\max} = 100$ packets and the slot duration is $20msec$. We set $\alpha_{n,k}(t) = 1$ in our experiments as our applications do not change the rate as explained later in this section. The second task of this thread is to actually do the computation tasks related to the application. In our experiments, the computation block counts the bytes in the packets. In particular, similar to the pilot study in the introduction, $\mathcal{O}(1)$, $\mathcal{O}(n)$, and $\mathcal{O}(n^2)$ correspond to (i) no counting, (ii) counting every byte in a packet once, and (iii) counting every byte in a packet n times, respectively.

The third thread transfers packets from $Q_{n,k}$ to $Z_{n,k}$ using the energy control algorithm in (9), where we set $E_{n,k}^{\max}$ depending on the battery level of the device. For example, if the battery level is below some threshold, $E_{n,k}^{\max}$ is limited. We evaluated different configurations in our experiments as we explain later. The slot duration is again set to $20msec$.

The final thread transfers packets from $Z_{n,k}$ to application layer if $n = k$, or transmits to node k if $n \neq k$. In the second case, *i.e.,* if $n \neq k$, the number of packets in TCP socket is checked at every time slot, where the time slot duration is $20msec$. If it is below a threshold of 500 packets, then 100 packets are removed from $Z_{n,k}$ and inserted to the TCP socket to be transmitted to node k.

When node n receives packets from node k, it directly passes the packets to the application layer as illustrated in Fig. 3(c), because these packets are the ones that are already processed by node k. If node n is both a helper and a receiver device, it runs all the threads explained above in addition to the receiving thread from node k (illustrated in Fig. 3(c)).

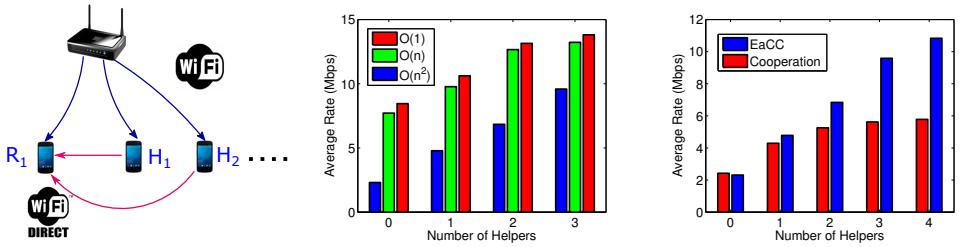
Information Exchange: Our implementation is lightweight in the sense that it limits control information exchange among mobile devices. The only control information that is transmitted in the system is $U_{n,k}$ from each mobile device to the source node. Each mobile device n collects $U_{n,k}$, $\forall k \in \mathcal{N}$, and transmits this information to the source node periodically, where we set the periods to $100msec$.

Connections: All the devices in the system including the source device, helpers, receivers, and helper+receiver devices are connected to each other using Wi-Fi Direct connections in our testbed. The source node is configured as the group owner of the Wi-Fi Direct group. We note that cooperation in this setup does not bring any benefit in terms of bandwidth utilization as all the links use the same transmission channel in a Wi-Fi Direct group. However, as we demonstrate later in this section, it brings benefit due to cooperative processing power and energy utilization, which is our main focus in this paper. Therefore, this setup (where all the devices are connected to each other using Wi-Fi Direct links) well suits to our evaluation purposes.

Test Environment: We conducted our experiments using our testbed in a lab environment where several other Wi-Fi networks were operating in the background. We located all the devices in close proximity of each other, and we have evaluated EaCC for varying levels of computational complexity, number of receivers, and number of helpers. Next, we present our evaluation results.

B. Results

We first consider a setup as shown in Fig. 4(a) which consists of a source device, one receiver (R_1), and one helper (H_1). Fig. 4(b) shows the average rate versus time graph for the setup shown in Fig. 4(a) when all three devices are Android



(a) System Model (b) Rate vs. Number of Helpers (c) Rate vs. Number of Helpers

Fig. 5. (a) System model consisting of a source device, one receiver, and multiple helpers. Wi-Fi is used between the source and the receiver device (R_1) and the helper devices ($H_1, \ldots$), while Wi-Fi Direct is used to connect the receiver devices with the helper devices. (b) EaCC: Average rate measured at receiver R_1 versus the number of helpers. (c) Average rate measured at receiver R_1 versus the number of helpers for EaCC and cooperation, when the complexity is $O(n^2)$.

based Nexus 7 tablets. The average rate is calculated as the average over 10 trials (with different seeds). The computation under consideration in this experiment is $O(n^2)$, which counts the number of bytes in a packet n times, n is the packet size. As can be seen, if there is no cooperation, the rate measured at R_1 is on the order of 1.5Mbps. On the other hand, EaCC increases the rate to almost 3Mbps. This means that helper device H_1 helps the receiver device R_1 process the packets in EaCC. In this setup, EaCC doubles the rate as compared to no-cooperation, which is a significant improvement.

For the same setup in Fig. 4(a), we also evaluate the impact of energy control part of EaCC on the average rate performance. In particular, Fig. 4(c) shows the average rate versus battery level at the receiver device R_1. In these results, we used Android based Nexus 5 smartphones. The average rate is calculated as the average over 10 trials (with different seeds). The computation under consideration in this experiment is $O(n^2)$, which counts the number of bytes in a packet for each byte in the packet. We consider that if the battery level of a device reduces below 40% threshold, then energy credits are not generated for the processing of the received packets. This makes $Q_{n,k}$ large over time, and after some point no packets are transmitted to that device for the processing task. In Fig. 4(c), when the battery level of R_1 reduces below 40%, then it stops receiving packets for processing. If there is no cooperation, then the rate towards R_1 reduces to 0. On the other hand, with EaCC, the rate is still higher than 0 thanks to having helper. The helper device with larger energy level (for the sake of this experiment), receives packets from the source, processes them, and forwards them to R_1, which receives already processed data. After 40% threshold, both EaCC and no-cooperation improve, because R_1 starts processing packets. This result shows the importance of energy-awareness in our cooperative computation setup.

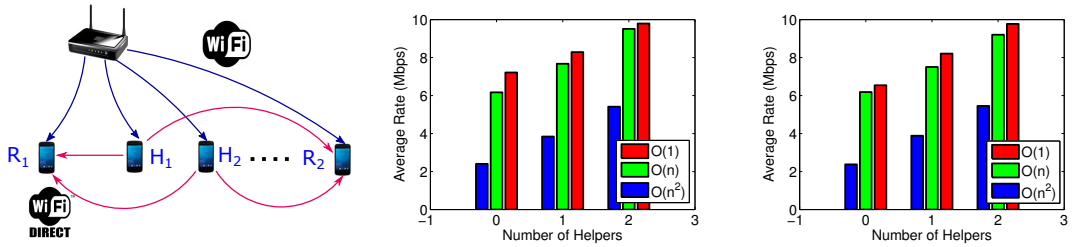
Now, we consider the impact of the number of helpers to overall rate performance. In particular, we develop a setup shown in Fig. 5(a), where there is one source, one receiver, and a varying number of helpers. In this setup, the source device, receiver, and the first two helper devices are Nexus 5 smartphones, while the other helpers are Nexus 7 tablets. Fig. 5(b) shows the average rate (averaged over 10 seeds) when EaCC is employed versus the number of helpers for different computational complexities such as $O(1)$, $O(n)$, and $O(n^2)$, where the processing task is counting the number of bytes in a packet. As expected, when complexity increases, the rate decreases. More interestingly, the increasing number of helpers increases the rates of all complexity levels. There are two reasons for this behavior. First, even if complexity level is low, *e.g.,* $O(1)$, processing power is still a bottleneck, and it can be solved by increasing the number of helpers. Note that after the number of helpers exceeds a value, the achievable rates saturate, which means that processing power is not a bottleneck anymore, but bandwidth is. The second reason is that receiving data over multiple interfaces increases diversity. In other words, when the channel condition over one interface (*e.g.,* between source and the mobile device) degrades, the other interface (*e.g.,* between two mobile devices) can still have a better channel condition.

In order to understand the real impact of processing power in a cooperative system, we tested both EaCC and cooperation (without computation and energy control) in the setup shown in Fig. 5(a). The results are provided in Fig. 5(c) when the complexity is $O(n^2)$. As can be seen, while EaCC significantly increases the rate with increasing number of helpers, cooperation slightly increases the rate (due to diversity). The improvement of EaCC over cooperation is as high as 83%, which is significant.

Finally, we consider a scenario that there are multiple receivers interested in different files. Fig. 6(a) shows the system model with one source, two receivers, and multiple helpers. In this setup, the source, two receivers, and the first helper is Android based Nexus 5 smartphone, while the rest of the helpers are Nexus 7 tablets. Fig. 6(b) and (c) show the average rate (averaged over 10 seeds) measured at R_1 and R_2 when EaCC is employed with respect to the increasing number of helpers, respectively. Similar to previous setups, $O(1)$, $O(n)$, and $O(n^2)$ correspond to different computational complexities, where the processing task is counting the number of bytes in a packet. As can be seen, the measured rate at both R_1 and R_2 increases with increasing number of helpers. This shows that our EaCC algorithm successfully accommodates multiple flows and receivers.

VII. Conclusion

We considered that a group of cooperative mobile devices, within proximity of each other, (i) use their cellular or Wi-Fi (802.11) links as their primary networking interfaces, and (ii)



(a) System Model (b) Rate at R_1 vs. Number of Helpers (c) Rate at R_2 vs. Number of Helpers

Fig. 6. (a) System model consisting of a source device, two receivers, and multiple helpers. Wi-Fi is used between the source and the receiver devices (R_1, R_2) and the helper devices ($H_1, \ldots$), while Wi-Fi Direct is used to connect the receiver devices with the helper devices. (b) EaCC: Average rate measured at receiver R_1 versus the number of helpers. (c) EaCC: Average rate measured at receiver R_2 versus the number of helpers.

exploit their D2D connections (Wi-Fi Direct) for cooperative computation. We showed that if mobile devices cooperate to utilize their aggregate processing power, it significantly improves transmission rates. Thus, for this scenario, we developed an *energy-aware cooperative computation* framework to effectively utilize processing power and energy. This framework provides a set of algorithms including flow, computation and energy controls as well as cooperation and scheduling. We implemented these algorithms in a testbed which consists of real mobile devices. The experiments in the testbed show that our *energy-aware cooperative computation* framework brings significant performance benefits.

REFERENCES

[1] "Cisco visual networking index: Global mobile data traffic forecast update," 2014-2019.

[2] "Ericsson mobility report," February 2015.

[3] "Wi-fi direct," http://www.wi-fi.org/discover-and-learn/wi-fi-direct.

[4] "Android," http://developer.android.com/develop/index.html.

[5] "Nexus tech specs," https://support.google.com/nexus/answer/6102470?hl=en.

[6] A. Singh, Y. Xing, and H. Seferoglu, "Cooperative compuration in device-to-device networks." [Online]. Available: http://nrl.ece.uic.edu and via [cs.NI] arXiv:1602.04400

[7] T. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. Wong, J. Schulz, M. Samimi, and F. Gutierrez, "Millimeter wave mobile communications for 5g cellular: It will work!" *Access, IEEE*, vol. 1, pp. 335–349, 2013.

[8] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with h.264/avc: tools, performance, and complexity," *Circuits and Systems Magazine, IEEE*, vol. 4, no. 1, pp. 7–28, First 2004.

[9] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/avc baseline profile decoder complexity analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 704–716, July 2003.

[10] J. K. Sundararajan, D. Shah, M. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets tcp: Theory and implementation," *Proceedings of the IEEE*, pp. 490–512, March 2011.

[11] P. Vingelmann, P. Zanaty, F. Fitzek, and H. Charaf, "Implementation of random linear network coding on opengl-enabled graphics cards," in *Wireless Conference, 2009. EW 2009. European*, May 2009, pp. 118–123.

[12] H. Shojania, B. Li, and X. Wang, "Nuclei: Gpu-accelerated many-core network coding," in *INFOCOM 2009, IEEE*, April 2009, pp. 459–467.

[13] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[14] M. Chiang, S. T. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: a mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, January 2007.

[15] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, T. Korakis, and L. Tassiulas, "Bits and coins: Supporting collaborative consumption of mobile internet," in *Proc. IEEE Infocom*, Hong Kong, April 2015.

[16] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," April 2014, technical report - arxiv:1310.0720v6[cs.GT].

[17] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proc. of ACM MobiHoc*, Hong Kong, May 2008.

[18] C. Boldrini, M. Conti, and A. Passarella, "Exploiting users' social relations to forward data in opportunistic networks: The hibop solution," in *Proc. of Pervasive and Mobile Computing*, October 2008.

[19] L. Keller, B. Cici, A. Le, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: Cooperative video streaming on smartphones," in *Proc. of ACM MobiSys*, June 2012.

[20] H. Seferoglu, L. Keller, A. Le, B. Cici, C. Fragouli, and A. Markopoulou, "Cooperative video streaming on smartphones," in *Proc. Allerton*, 2011.

[21] M. Ramadan, L. E. Zein, and Z. Dawy, "Implementation and evaluation of cooperative video streaming for mobile devices," in *Proc. of IEEE PIMRC*, Cannes, France, September 2008.

[22] S. Li and S. Chan, "Bopper: wireless video broadcasting with peer-to-peer error recovery," in *Proc. of IEEE ICME*, Beijing, China, July 2007.

[23] J. Chesterfield, R. Chakravorty, I. Pratt, S. Banerjee, and P. Rodriguez, "Exploiting diversity to enhance multimedia streaming over cellular links," in *Proc. of IEEE INFOCOM*, March 2005.

[24] ——, "A system for peer-to-peer video streaming in resource constrained mobile environments," in *Proc. of ACM U-NET*, December 2009.

[25] S. Ioannidis, A. Chaintreau, and L. Massoulie, "Optimal and scalable distribution of content updates over a mobile social network," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[26] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, "Cellular traffic offloading through opportunistic communications: a case study," in *Proc. of ACM Workshop on Challenged Networks (CHANTS)*, Chicago, IL, September 2010.

[27] R. K. Lomotey and R. Deters, "Architectural designs from mobile cloud computing to ubiquitous cloud computing - survey," in *Proc. IEEE Services*, Anchorage, Alaska, June 2014.

[28] T. Penner, A. Johnson, B. V. Slyke, M. Guirguis, and Q. Gu, "Transient clouds: Assignment and collaborative execution of tasks on mobile devices," in *Proc. IEEE GLOBECOM*, Austin, TX, December 2014.

[29] M. Satyanarayanan, S. Smaldone, B. Gilbert, J. Harkes, and L. Iftode, "Bringing the cloud down to earth: Transient pcs everywhere," in *MobiCASE'10*, 2010, pp. 315–322.

[30] E. Miluzzo, R. Caceres, and Y. Chen, "Vision: mclouds - computing on clouds of mobile devices," in *ACM workshop on Mobile cloud computing and services*, Low Wodd Bay, Lake District, UK, June 2012.

[31] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in mul- tihop radio networks," *IEEE Trans. on Automatic Control*, vol. 37, no. 12, December 1992.

[32] ——, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. on Information Theory*, vol. 39, no. 2, March 1993.

[33] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE Trans. on Networking*, vol. 16, no. 2, April 2008.

[34] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE/ACM Transactions on Information Theory*, vol. 34, no. 5, March 2000.

Device-to-Device Mobile Data Offloading for Music Streaming

Sylvia T. Kouyoumdjieva and Gunnar Karlsson
ACCESS Linnaeus Center, School of Electrical Engineering
KTH Royal Institute of Technology, Stockholm, Sweden
Email: {stkou, gk}@ee.kth.se

***Abstract*—Device-to-device communication (also referred to as opportunistic networking) is considered a feasible means for offloading mobile data traffic. Due to the sporadic nature of contact opportunities, applications in the domain of device-to-device communication are assumed to be delay-tolerant, with content delivery deadlines being in the order of hours. However, predictions suggest that by 2020 more than 75% of the traffic volumes at mobile operators will be generated by multimedia contents which is often seen as data served in real-time. In this paper we explore how the concept of opportunistic networking can be used for dissemination of real-time streaming contents for users in urban environments without degrading quality of experience. We first present a general framework for offloading multimedia data that is organized in terms of playlists, and we then investigate the performance of the framework in realistic urban environments using the music streaming service Spotify as a use-case. Our results show that it is feasible to use opportunistic device-to-device communication in the context of music streaming. We demonstrate that the system performance is insensitive to a number of parameters such as playlist length distribution, and initial content availability distribution, however it exhibits sensitivity towards the amount of requested data and the node density.**

***Index Terms*—mobile data offloading, device-to-device communication, opportunistic networking, music streaming, Spotify.**

I. Introduction

The proliferation of mobile devices has changed tremendously the way in which people consume information. The huge amounts of data delivered to mobile users on an everyday basis requires mobile operators to face the challenge of increased traffic volumes in their networks. In fact, predictions are that by 2020 mobile operators will experience around 24 EB of monthly mobile data traffic. Approximately 75% of this traffic is expected to be generated by multimedia contents [1]. Mobile operators are thus exploring different possibilities for offloading traffic volumes to alternative networks, and one of the promising solutions that has been proposed is device-to-device (also called opportunistic) communication. Opportunistic communication allows devices in proximity to exchange data directly with each other without relying on infrastructure.

Opportunistic communication has been perceived as a means of disseminating and offloading delay-tolerant contents characterized with content delivery deadlines in the order of hours such as news or software updates. However with the increasing amounts of multimedia contents such as music and video delivered to mobile devices, offloading solely delay-tolerant contents may not be enough to reduce the traffic volumes at mobile operators. Thus, we here question whether the concept of device-to-device opportunistic communication could also be applicable for disseminating data with much shorter deadline requirements, such as streaming data. Only few studies [2], [3], [4] evaluate direct device-to-device communication as an alternative for delivering on-demand streaming to mobile devices, however the solutions rely on the presence of infrastructure for supporting the dissemination process. Instead, we evaluate the feasibility of an infrastructureless solution for offloading streaming data while maintaining the quality of experience for the end user.

The contributions of this work are two-fold. We first introduce a framework for disseminating real-time streaming contents in an opportunistic manner. We then evaluate the performance of the framework via extensive trace-driven simulations based on realistic mobility traces that recreate mobility patterns of urban users. We base our evaluation on the popular music streaming service Spotify as a use-case. The rationale behind choosing a music streaming service as a use-case is the length of the flows. Music streams are long-lived (in the order of hours) and although they do not require high data rates, due to their duration they result into high traffic volumes. Our results show that opportunistic device-to-device communication is a viable means for offloading streaming data. The system performance exhibits high sensitivity towards the node density and the amount of requested data, but it is insensitive towards the distribution of the playlist length and the initial content availability.

The rest of this paper is structured as follows. In Section II we present our proposed general framework for offloading multimedia contents via device-to-device communication. Section III presents a popular music streaming service, Spotify, which we further use as a case study for evaluating the framework. Sections IV and V summarize the evaluation scenario and results, while Section VI discusses the related work and positions our proposal with respect to it. Finally, we conclude in Section VII.

II. Opportunistic Dissemination and Streaming

Opportunistic device-to-device communication allows nodes equipped with mobile devices to exchange data with one

ISBN 978-3-901882-83-8

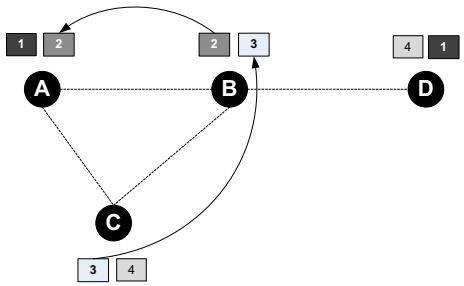


Fig. 1. An example with four nodes {A, B, C, D} interested in four content items {1, 2, 3, 4}. Each node has one content item in its cache (left rectangle) and one content item in its requested playlist to obtain in the nearest future (right rectangle). Dashed lines show nodes in direct communication range; solid lines show possibilities for content exchange.

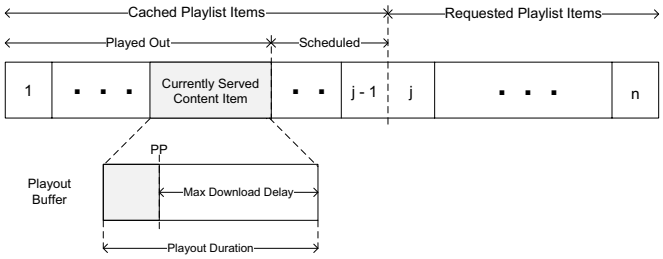


Fig. 2. The cached playlist holds items that are already available on the device; these can be either items that have been played by the device, or items that have been downloaded for future usage. The requested playlist holds items that still need to be downloaded. The playout point is denoted with PP.

another when in direct communication range. Data exchange is only possible if nodes share one or more common interests. An interest is expressed in the form of a subscription to a specific content category through a publish/subscribe interface exposed by an opportunistic content dissemination system [5]. Nodes encounter other peers, and are consecutively able to discover and download contents stored on those peers, as they move through an area. However, without keeping track of the mobility pattern of each and every node in the system, it is not possible to predict beforehand neither when any two devices with shared interest would be in direct communication range, nor how long the duration of their contact would be. Due to this unpredictability, opportunistic communication is mostly proposed as an offloading solution for delay-tolerant content, and is often considered inappropriate for sharing real-time contents such as streaming data.

Streaming data *is* provided in real-time when considering the data that is delivered to a device. However, shifting the viewpoint towards contents changes the definition from real-time data into *data with tight delay constraints*. We define contents to have tight delay constraints if its playout is in real-time but its delivery to the requesting application happens within some predefined delay boundaries. Contrary to traditional on-demand data delivery, the delay boundaries in this case are more generous. Let us illustrate this concept with an example, Fig. 1. Four mobile nodes {A, B, C, D} are interested in obtaining four content items {1, 2, 3, 4} constituting a *playlist*. The dashed lines show direct communication links between nodes. At the beginning, all playout buffers are empty. Thus, each of the nodes begins by streaming one of the items of interest from a server via the cellular network and playing it out; node A streams item 1, node B streams item 2, etc. Since all nodes are interested in all four of the content items, the application at each node is aware both of the currently played content item, as well as of other content items that are not available on the device but are part of the playlist; we call this the *request playlist*. In this example, the request playlist of node A consists of item 2, the request playlist of node B consists of item 3, etc. Normally node A would request item 2 from the server and stream it in real-time once it is done streaming item 1. However, if node A's application is aware of the items in its request playlist, it may as well attempt to obtain these items in advance in alternative ways, for example via opportunistic device-to-device communication, *before* they are requested by the application. Then, once the streaming of item 1 is over, item 2 can be played seamlessly from a local cache. In our example, node A has item 2 in its request playlist, while node B which is a neighbor of node A already has item 2 which it has obtained via streaming; node B could consecutively share item 2 upon request with node A.

In the context of delivering multimedia to mobile devices, a node can either *stream* the data in real-time directly from a server via the cellular network, or it can *download* the data from a peer via an opportunistic contact. Contents available in the cache of a node is then *played out* whenever the application requests it. Observe that play out is oblivious whether the data source is local (the device's cache) or remote (the server).

Fig. 2 illustrates the general concept of cached and requested playlist items. A *cached* playlist item is already available on the device. Observe that we do not specify how the cached item has been made available to the device. For instance, it could have been streamed via a cellular network, or downloaded via an opportunistic contact. A *requested* playlist item is a content item that the device is expected to provide to the application in the future. A special case of a cached item is the currently served content item, i.e. the item that is being played out to the requesting application. The currently served content item defines the maximum download delay as the time until the next play out. In case that the currently consumed content item is the last item in the cached playlist, the maximum download delay measures how long the node could search new content items in neighboring devices *before* its buffer under-flows. Algorithm 1 presents the framework for offloading streaming data with tight delay constraints. As long as the request playlist of a node is not empty (line 3), the node actively searches for peers in its vicinity that could provide it with useful contents. Upon downloading a content item from a neighbor, the node stores the item in its scheduled playlist, and removes it from the request playlist (lines 7 and 8). Whenever the playout buffer reaches the end of the currently consumed content item (line 10), the requesting application checks whether there are any scheduled content items; if this

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Algorithm 1 Framework for Offloading of Streaming Data

1: $\mathcal{S} \leftarrow$ set of scheduled content items
2: $\mathcal{R} \leftarrow$ set of requested content items
3: **while** $\mathcal{R}$ is not $\varnothing$ **do**
4: search for peers in vicinity
5: **if** available peer with shared interest **then**
6: download content item
7: add content item to $\mathcal{S}$
8: remove content item from $\mathcal{R}$
9: **end if**
10: **if** playout buffer is $\varnothing$ **then**
11: **if** $\mathcal{S}$ is $\varnothing$ **then**
12: **if** replay **then**
13: replay cached content item
14: **else**
15: stream content item from server
16: **end if**
17: **else**
18: play out next content item
19: **end if**
20: **end if**
21: **end while**

is the case, the first of the scheduled content items is played out from the local cache (line 18). However, if the scheduled content items playlist is empty, the requesting application has two options: either to replay a locally cached content item that has already been played out before (line 13) or to stream a new content item from the server via the cellular network (line 15).

The playout system in a device can be described as a G/G/1 queueing system, Fig. 3. Nodes arrive in the communication range of an observer node j with mean arrival rate λ. However, only a subset of these nodes could be useful to the observer node. A contact is useful if it can deliver contents of interest to the observer node while in its communication range. Useful contacts occur at a content solicitation rate $\lambda' < \lambda$, and can be described with a distribution of useful inter-contact times $f(t)$. The distribution of the useful inter-contact times is defined by the underlying node mobility, as well as the popularity of the content items in the requested playlist of the observer node. The duration of the currently served content item τ in node j defines the service time. The service time follows a distribution $g(\tau)$. Opportunistic mobile data offloading for streaming services is then defined as feasible *if* the mean inter-arrival time of useful nodes to node j is larger than the mean service time for node j, $\mathbb{E}[f(t)] > \mathbb{E}[g(\tau)]$. Contrary to traditional queueing theory, the main objective for the system is to operate in a saturated state.

III. Music Streaming with Spotify

Spotify is a popular online peer-assisted music streaming service [6]. Spotify offers an extensive music catalog with more than 20 million music tracks to desktop, mobile and web users. In the wired domain, users can stream music directly from the Spotify servers or via peers who have already downloaded the track and cached it on their devices. It has been shown that approximately 40% of all data delivered to users is provided by peers instead of by the server. In the wireless domain, however, users are currently only allowed to download tracks directly from the Spotify servers. Recently Spotify announced that the usage of their service on mobile devices has surpassed the usage of desktop and web clients, with more than 50% of users streaming music on their smartphones or tablets [7]. This shift in usage significantly increases the traffic volumes on the servers as well as on the mobile operator, and would ultimately require a shift in the way data is provisioned to devices in the wireless domain.

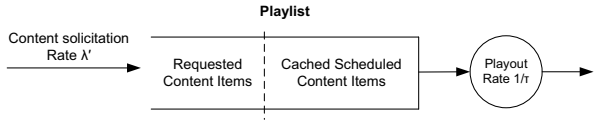


Fig. 3. The playout system in a device represented as a G/G/1 queueing system.

```
<track href="spotify:track:5eCgNATwXgRc4mZx9NymGJ">
   <name>Waiting For Love</name>
   <artist href="spotify:artist:1vCWHaC5f2uS3yhpwWbIA6">
      <name>Avicii</name>
   </artist>
   <album href="spotify:album:0LUr5Q06EQu7QIid7cACFU">
      <name>Waiting For Love</name>
      <released>2015</released>
      <availability>
         "AD", "AR", ... "TW", "UY"
      </availability>
   </album>
   <track-number>1</track-number>
   <length>228.750000</length>
   <popularity>96</popularity>
</track>
```

Listing 1. Sample data structure of a Spotify track. Some parameters are omitted in order to improve readability.

A. Spotify Data Analytics

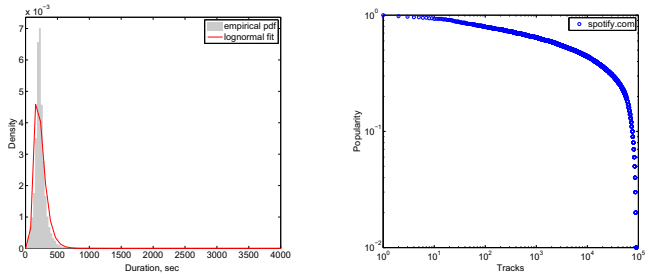
Spotify provides an application programming interface (API)[1] which allows developers to access and extract meta data about tracks, artists as well as users and their playlists. Searching through Spotify's database is performed by a simple GET request:

`GET http://api.spotify.com/v1/search?q=text&key`$_i$`=val`$_i$

Here, `text` denotes a free text, for instance a name of a track, while `key`$_i$ and `val`$_i$ are additional key-value filtering options for improving the search results. For example, in this section we make use of the `market` key which expects an ISO 3166-1 alpha-2 country code as a value to narrow down search results to a specific country. Another popular filtering key option is `type` which can take values of `track, artist` and `album`.

The data provided by the API in response to a GET search request is presented in the form of a JSON object, and in essence it consists of a collection of key-value pairs associated with a particular request. A sample data structure

[1]Spotify's API is available at http://developer.spotify.com/web-api/



(a) (b)

Fig. 4. Statistics collected from 100 000 tracks on Spotify. (a) Distribution of track durations. (b) Distribution of track popularity.

corresponding to a search of a popular track is presented in Listing 1; the format is converted into XML for the sake of readability. The track element is the highest level element in the structure, and it contains various meta data associated with the particular track, such as the length of a track (in milliseconds), the associated artist and album, as well as the track availability across the world. Each track has a unique identifier in the form of a URI which is provided by the `href` key.

To obtain a better understanding of the parameters of data distributed by the Spotify service, we implemented a simple crawler in Python and collected statistics of 100 000 tracks out of the Spotify catalog. Fig. 4(a) presents the distribution of track durations. Most tracks have durations of few minutes, with the mean track duration being approximately 225 s. However we see that few tracks exhibit much longer durations; these are for example classical music pieces. None of the general probability distributions is able to fully describe the distribution of track durations however the lognormal distribution gives the closest fit.

Spotify also provides their own estimate of the popularity of tracks as a value between 0 and 100, with 100 denoting the most popular tracks. Popularity is estimated with respect to the total number of times a track has been played, as well as how recently it has been played. Fig. 4(b) illustrates the popularity distribution of 100 000 tracks plotted on a loglog scale. (We here represent popularity with values between 0 and 1 on the y-axis to improve readability.) Only few tracks have a high popularity score. Those are the tracks that could potentially be solicited by neighboring peers instead of streamed directly from the Spotify server in the wireless domain. We therefore further focus on evaluating only high popularity tracks. However we should note that when considering tracks for disseminating via opportunistic device-to-device contacts, we should not look into track popularity on global scale. Instead, since device-to-device communication allows data exchange among devices in proximity, we should attempt to describe data popularity in a smaller and more local scope, i.e. the scope of a country or even an area in a country. Spotify currently provides data at the scope of a country in its availability tag.

Fig. 5 shows the popularity distribution of tracks currently present in the Top 50 chart for three European countries:

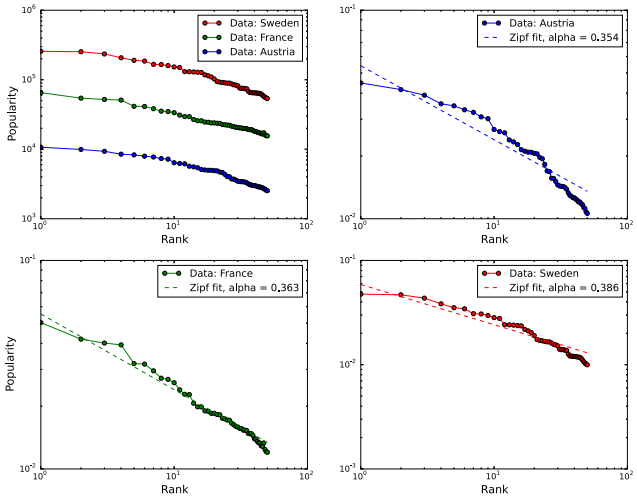


Fig. 5. The popularity distribution of the top 50 tracks per country can be described with a Zipf distribution. Popularity is calculated based on the number of times a track has been played in the course of a week. Although tracks tend to have different popularity across countries, the parameter describing the Zipf distribution does not vary significantly.

Austria, France and Sweden. (We take into account only data from European countries instead of the USA or China due to the fact that the area is smaller and the statistics represent a more localized view of the track popularity.) The graph plots the popularity of tracks in terms of the times a track has been played with respect to the position of the track in the chart. Users in different countries exhibit different usage patterns of the Spotify service, with the service being most commonly used in Sweden, and least used in Austria (upper left in Fig. 5). We then fit the popularity curve into a Zipf distribution, and we estimate the parameter α for each of the three datasets. It is interesting to see that although the usage statistics per country vary significantly (the most popular song in Sweden is played more than 250 000 times, while in Austria – less than 10 000 times), the Zipf distribution that describes the popularity exhibits similar behavior (with $\alpha \in [0.354, 0.386]$).

Finally, we provide a rough quantitative measure of the maximum offload that could be achieved if popular tracks are to be disseminated via opportunistic contacts instead of streaming the contents directly from the Spotify servers. Table I shows the amount of data downloaded from the server when the duration of each song has a mean of 225 s, the audio stream is encoded with Ogg Vorbis q9 with 320 kbps (the typical encoding scheme for premium users of Spotify). Currently, the top 50 most popular tracks amount to hundreds of terabits of data downloaded from a server, and traversing a mobile network which could potentially be offloaded to alternative networks.

B. Spotify and Opportunistic Communication

In order to be able to examine the performance of a music streaming service such as Spotify in the opportunistic domain, we first need to define how the current protocols and data formats would fit into the context of device-to-device

TABLE I
TOP 50 CHART: TOTAL WEEKLY DOWNLOADS AND MAXIMUM OFFLOAD CAPACITY WHEN AUDIO STREAMS ARE ENCODED WITH OGG VORBIS Q9 AT 320 KBPS.

Country	Total downloads (millions)	Offload capacity (terabits)
Austria	0.2 M	16 Tb
France	1.3 M	80 Tb
Sweden	5.4 M	380 Tb

communication. We thus present a sample publish/subscribe framework for opportunistic networks. Similar to the publish/subscribe interface exposed currently by Spotify to its users [8], in the opportunistic domain users should be able to express interest in the tracks they wish to listen to. However, while in the wired domain a user is able to subscribe to a playlist or to an artist or to another user, in the wireless domain such a fragmentation in subscriptions would lead to an underutilization of the opportunistic network. For instance, a node is less probable to encounter another peer which is subscribed to the same playlist, however a user has higher chances of encountering another node which is subscribed to the same track. The reason is that tracks may exist in multiple playlists, and they are always defined by a single unique identifier.

IV. EVALUATION SCENARIO

A. Mobility scenario

In order to realistically recreate pedestrian mobility, we use the Walkers traces [9] captured in Legion Studio [10], a commercial simulator initially developed for designing and dimensioning large-scale spaces via simulation of pedestrian behaviors. Its multi-agent pedestrian model is based on advanced analytical and empirical models which have been calibrated by measurement studies. Each simulation run conducted in Legion Studio results in a mobility trace file, containing a snapshot of the positions of all nodes in the system every 0.6 s.

Fig. 6 presents an urban outdoor scenario considered in our evaluation; the outdoor scenario recreates an actual part of downtown Stockholm. The scenario consists of a grid of interconnected streets with lengths varying between 20 m and 200 m. Each street has a width of 2 m which is representative of a sidewalk. In terms of cellular coverage, we assume that the area of the outdoor scenario corresponds to the area of a single cell of a mobile operator. Nodes enter into the urban area according to an arrival rate λ via one of the fourteen passages that connect the area to the outside world, and roam around the area until they reach an exit passage. The mean sojourn time of nodes is approximately 295 s. The active area of the scenario is 5872 m^2; observe that the active area defines the area through which nodes can actually move, i.e., the streets. Throughout their lifetime nodes are constantly moving in the observed area, therefore the scenario can be characterized as a high mobility scenario. More details on the scenario can be found in [11].

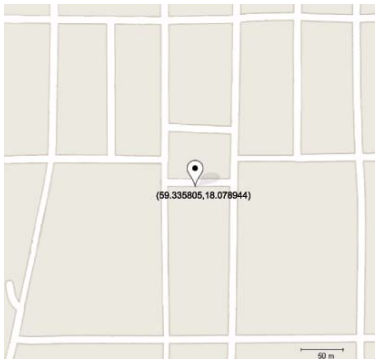


Fig. 6. Urban scenario: a grid of streets representing a part of an actual downtown area in Stockholm.

B. Simulation Setup

In our evaluation scenario we assume that all nodes carry mobile devices on which the Spotify application is installed. Communication between nodes may occur when they are in direct communication range; we set the range to be 10 m. Since most technologies dedicated to device-to-device communication, such as Wi-Fi Aware [12] and LTE-Direct [13], are not yet mature, we here do not focus on evaluating the performance of opportunistic mobile data offloading with respect to a specific underlying technology. Instead we only assume that the data rate between nodes is regulated to 10 Mbps.

Nodes have some contents preloaded into the caches of their devices, and while this content is played out, they attempt to obtain one or more tracks from their request playlist. Observe that from the viewpoint of the Spotify service, nodes may be subscribed to different playlists. However, as long as these playlists contain the same track, opportunistic device-to-device data exchange is possible. In this work we only evaluate the performance of the system with respect to these overlapping popular tracks as part of the request playlist. We believe that such an evaluation is realistic based on the track popularity distribution exhibited in the previous section. We assume that a total of $\mathcal{N}$ tracks, belonging to one or more Spotify playlists, are available in the area, and nodes may be subscribed to a set of them, or to all of them. Based on the data collected from Spotify in Section III we assume that the mean duration of a track is 225 s, with a standard deviation of 30 s. The popularity of tracks follows a Zipf distribution with parameter $\alpha = 0.368$; observe that the value of α is chosen in correspondence with the results obtained in Section III. For each simulation run we first remove the transient phase, and we then collect statistics of 1000 nodes in steady state. For all results the mean values are plotted with a 95% confidence interval.

C. Performance metrics and configurations

We focus on the following three metrics in our evaluation:

- **User satisfaction**: The user satisfaction is a measure of the percentage of users who are able to obtain *at least* one track from their request playlist via an opportunistic contact with a neighboring node before the maximum

download delay associated with the currently consumed content item is reached.

- **Offload ratio**: The offload ratio is a measure of the amount of data obtained via opportunistic contacts with respect to the total amount of data in the request playlist of a node. We only account for downloads from peers that lead to transfer of an entire track.
- **Inter-download time**: The inter-download time defines the interval between two consecutive encounters with nodes that are able to provide contents of interest to a given node. The ability of a node to provide contents of interest is in turn defined by the local subscriptions and the contact duration. Contacts with duration shorter than the minimum transfer time for a track are discarded.

Although energy consumption is an important metric in the context of device-to-device communication, we do not address it in this work; instead we refer the reader to [14] where we present power-saving algorithms for opportunistic mobile data offloading.

V. Results

A. Effect of Request Window Size

Let us assume that all nodes are interested in obtaining a playlist, constituting a total of $\mathcal{N} = 50$ tracks. Nodes that enter the area have one track pre-loaded in their cache, and are initially interested in obtaining k tracks from the request playlist. A request window of size $k = 1$ denotes that a node is interested in obtaining the next item on its playlist. We refer to this type of playlist as a *preset* playlist. A preset playlist may for instance contain a collection of podcast episodes which need to be reproduced in the correct order at the end device. A node that has a request window of size $k > 1$ is interested to obtain any of the next k tracks on its playlist; there is no need to preserve the order of tracks on the playlist, hence the playlist is *random*. An example of a random playlist is a playlist of independent music tracks which can be played in any order. Note that in this study we do not consider a scenario in which nodes can interact with the playlist, i.e., we assume that nodes do not fast-forward or skip a track during their lifetime in the system.

Fig. 7 shows the effect of the request window size on the offload ratio for the urban scenario with two different arrival rates. The offload ratio increases both with the size of the request window and with the density of nodes in the area. In fact, when a critical mass of nodes is present in the area, even at relatively small values of the request window $k = 5$, the offload ratio is approximately 90%. However, the offload ratio should not be considered in isolation; instead it should be coupled with the user satisfaction. A user is *satisfied* when its buffer does not underflow, i.e. when it is able to obtain at least one track out of its request playlist before the maximum download delay is reached. Observe that even if the offload ratio is low, the user satisfaction may be much higher (Table II) even at low densities. The reason is that the offload ratio provides a measure with respect to the overall size of

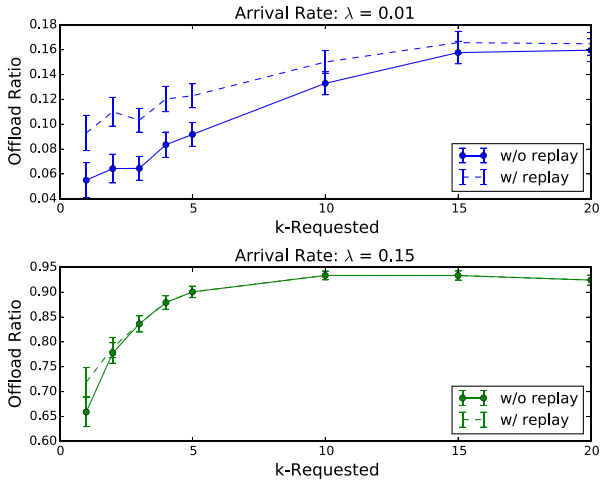


Fig. 7. Effect of the request window size for $\lambda = \{0.01, 0.15\}$ nodes/s on the offload ratio with and without replay. Observe the difference in the maximum value on the the y-axis.

TABLE II
User satisfaction with respect to the request window size.

Arrival rate	Request Window Size					
	1	2	3	4	5	10
λ = 0.01 n/s	6%	11.6%	16.9%	25%	31.9%	60.5%
λ = 0.15 n/s	66.9%	91.1%	97.3%	99%	100%	100%

the request window (thus the whole playlist), while the user satisfaction provides a measure for the momentary state of the node (what is to be played next).

Nodes that do not download a track before the maximum download delay is reached have two options: either to stream the track directly via the cellular network, or to replay one of the previous tracks that are already available in the device cache. The first strategy gives us the lower bound of offload ratio - no node is willing to replay tracks; the second strategy - the upper bound of the offload ratio - all nodes prefer to replay tracks rather than to download contents via the cellular network. Fig. 7 shows that at low densities, replaying tracks allows for a marginal improvement of the offload ratio at low values of the request window size, however as the request window size increases the offload ratio achieved with and without application of a replay strategy becomes comparable. At higher densities the replay strategy is almost never used since nodes have enough contact opportunities to download fresh contents.

B. Effect of Node Density

Fig. 8 illustrates the effect of the node density on the system performance in terms of offload ratio. We vary the arrival rate of nodes in the area from $\lambda = 0.01$ n/s to $\lambda = 0.15$ n/s for each entry point into the observed area, and we evaluate the offload ratio for two values of the requested playlist size, $k = 1$ (preset playlist) and $k = 10$ (random playlist). The results in Fig. 8 confirm that the system performance improves

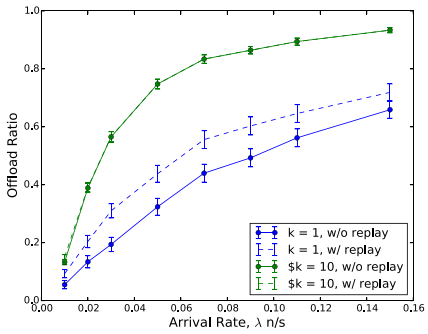


Fig. 8. Effect of node density on the offload ratio for two different values of the request window size, $k = 1$ (preset playlist) and $k = 10$ (random playlist).

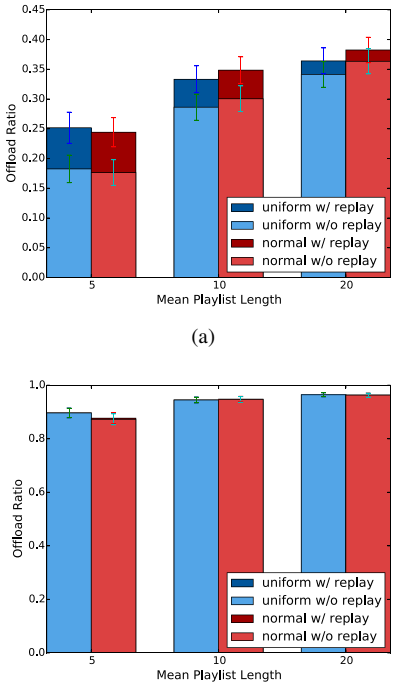
with the increase of node density. It is interesting that even at low node densities, track dissemination via device-to-device communication achieves relatively high offload ratio (more than 50% for $\lambda = 0.03$ n/s) when the size of the request playlist is large ($k = 10$) and the tracks from the request playlist can be downloaded in random order. Furthermore, a larger request window size results in faster increase of the offload ratio, especially for small values of the arrival rate. On the contrary, when the playlist is preset with $k = 1$, the change in the offload ratio between two consecutive values of the arrival rate is almost linear.

The results in Fig. 8 also illustrate that replay strategies are useful for preset playlists regardless of the node density. On the other hand, when the playlist is set to random with a large request window size, replaying tracks does not improve performance significantly even in sparsely populated scenarios ($\lambda = 0.01$ n/s).

C. Effect of Playlist Length

Users have different music preferences; thus nodes often would not be interested in obtaining all $\mathcal{N}$ tracks but only a subset $\mathcal{S} \subset \mathcal{N}$ of these tracks. In this section we evaluate whether the distribution of the mean playlist length of the subset $\mathcal{S}$ affects the system performance. Upon entering in the area, each node subscribes to a subset $\mathcal{S}$ of tracks distributed either uniformly or normally over all tracks in the original set $\mathcal{N}$. A subset of the playlist $\mathcal{S}$ is already available on each device upon entering the area. According to [6], approximately 50% of tracks that are requested by the Spotify application can be found locally in the cache of the requesting device. We thus assume that initially the cache of each device is populated with j tracks chosen uniformly at random from the $\mathcal{S}$ tracks in the playlist, with the mean value of j being $\mathbb{E}[j] = |\mathcal{S}|/2$ where $|\mathcal{S}|$ is the cardinality of the playlist set. The size of the request window of a node is then calculated as $|\mathcal{S}| \quad j$.

Fig. 9 shows how the distribution of the playlist length $|\mathcal{S}|$ affects the offload ratio for scenarios with different density. The offload ratio exhibits sensitivity towards the mean value of the playlist length, however it is not sensitive towards the distribution of the playlist length. Again, adopting replay



(a)

(b)

Fig. 9. Effect of playlist length distribution on the offload ratio with and without replay for (a) λ = 0.01 nodes/s, and (b) λ = 0.15 nodes/s. Observe the difference in the maximum value on the the y-axis.

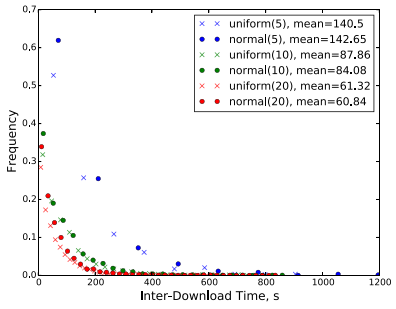
TABLE III
PERCENTAGE OF NODES THAT DO NOT OBTAIN A SINGLE TRACK DURING THEIR LIFETIME IN THE SYSTEM UNDER DIFFERENT DISTRIBUTIONS OF THE PLAYLIST LENGTH.

Playlist Length	Arrival Rate			
	λ = 0.01 n/s		λ = 0.15 n/s	
	Uniform	Normal	Uniform	Normal
$\mathbb{E}[j]$ = 5	76.5%	75.2%	32.9%	34.2%
$\mathbb{E}[j]$ = 10	50.2%	45.2%	18.5%	15.2%
$\mathbb{E}[j]$ = 20	31.7%	25.3%	10.4%	8.8%

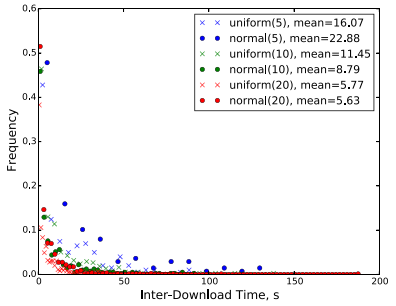
strategy is only beneficial for scenarios with low node densities, especially for short playlists. As the mean length of the playlist increases, and with the increase of participating nodes in the system, replaying tracks does not enhance the system performance.

We further take a look into the distribution of inter-download times, Fig. 10. Regardless of the node density, the shorter the length of the playlist $|\mathcal{S}|$, the longer the inter-download times. Furthermore, the distribution of playlist length does not significantly affect the distribution of inter-download times.

Finally, we evaluate the effect of playlist length distribution on content delivery. Table III shows the percentage of nodes that are not able to obtain a single track throughout their lifetime in the system. Similar to the offload ratio, the percentage of nodes that are not able to obtain contents via opportunistic contacts is strongly dependent on the mean length of the playlist as well as the node density in the area, however it is not affected by the actual distribution of the playlist length.



(a)



(b)

Fig. 10. Effect of playlist length distribution on the inter-download times for (a) λ = 0.01 nodes/s, and (b) λ = 0.15 nodes/s. Observe the difference in the maximum value on the the y-axis.

TABLE IV
AVERAGE INTER-DOWNLOAD TIMES FOR DIFFERENT DISTRIBUTION OF THE INITIAL CONTENT AVAILABILITY WITH MEAN $m = 25$ TRACKS.

Distribution	Arrival Rate	
	λ = 0.01 n/s	λ = 0.15 n/s
Deterministic	42.2 s	3.8 s
Uniform	45.2 s	3.8 s
Normal	45.2 s	3.8 s

D. Effect of Initial Content Availability

Lastly we examine the effect of initial content availability. We assume that all nodes are interested in obtaining all $\mathcal{N}$ tracks from a playlist, and that on average they have already stored half of the playlist in their local caches. We vary the distribution of cached tracks, and examine three different distributions: deterministic (exactly 50% of the items from the playlist are cached), uniform and normal. Table IV shows the average inter-download times for each of the distributions; we see that the inter-download times are not sensitive to the distribution of the initial contents available in the cache.

VI. Related work

Recent solutions for alleviating traffic load on cellular networks can be divided in two main categories: offloading to femtocells or existing Wi-Fi networks [15], [16], [4], and offloading through opportunistic device-to-device communication. Although the large body of work produced in the area of offloading cellular traffic pertain to femtocells and Wi-Fi networks, such approach is limited to possible deployments and the availability of Internet access. Using opportunistic communication for mobile data offloading does neither depend on available deployment, nor on Internet access, and has thus become a popular candidate for traffic offloading in recent years. A number of studies attempt to optimize the traffic volumes delivered to end users through opportunistic communication. In [17], Han et al. study a target-set selection problem for choosing initial data carriers in order to minimize the amounts of mobile data traffic. Lu et al. propose an opportunistic forwarding protocol for increasing the probability of data delivery [18]. Since mobile data offloading makes use of battery resources on battery-powered mobile devices, a body of work in the mobile offloading domain has been concentrated towards reducing energy consumption [14], [19]. However, all of the studies assume that opportunistic communication is used for offloading delay-tolerant contents. Instead, we evaluate whether opportunistic communication could be used for offloading data with tight delay constraints such as streaming multimedia organized in playlists.

Few works evaluate the performance of on-demand video streaming services in the presence of opportunistic contacts. In [2] Yoon et al. present a mobile peer-to-peer video-on-demand application which uses peers for delivering parts of a video stream to the video player. The application however relies heavily on the presence of a centralized scheduler which coordinates the data exchange among the peers, and it utilizes a complementary download link when contents cannot be fetched in real-time from nodes in the vicinity. In [3], Siris et al. present testbed experiments of multi-source mobile video streaming which exploits mobility and throughput prediction for prefetching video data in caches located at hotspots that the mobile will encounter and device-to-device communication to opportunistically obtain parts of a video from neighboring mobile devices. A delay-tolerant networking approach relying on multiple links and routing is applied to live-streaming by Morgenroth et al. in [20]. In contrast to these works, we do not aim to offload the currently served content item but instead we make use of opportunistic contacts to deliver contents that would be needed by the node in the near future. Furthermore, we do not rely on infrastructure for supporting the content delivery, neither do we incorporate routing for finding appropriate peers to deliver the stream. In [21], Keller et al. propose MicroCast, a system for cooperative video streaming among mobile devices in close proximity. MicroCast is however designed to operate for small number of participants in a static setting, and is thus not applicable for mobile scenarios with users on-the-go. Our approach is closest to the work of Ding et al. [4] in which they examine prefetching of non-live streaming contents. However, they rely on Wi-Fi infrastructure, on profiling user mobility patterns and on location reporting for determining the access points on which to offload future multimedia streams. Instead, our solution is purely based on opportunistic communication. Recently Jimenez et al. evaluated the effect on energy consumption when mobile devices are introduced as participants in Spotify's peer-to-peer overlay in the wired domain [22]. As opposed to them, we here evaluate a solution which creates

a separate overlay in the wireless domain and alleviates the communication with Spotify's servers which allows offloading mobile data from the operators' networks. Lastly, in [23] Danihelka et al. propose a hybrid architecture for video sharing based on cloud logic and device-to-device communication. Similar to us, the authors aim to offload the next to-be-played-out piece of video contents within a predefined deadline. The proposed architecture however requires a signalling feedback to assist the device-to-device content spreading. Instead, in our work we do not require nodes to provide feedback to the network, and the content dissemination is purely opportunistic.

VII. Conclusion

It is expected that by 2020 more than 75% of the traffic volumes at mobile operators will be generated by real-time multimedia services such as audio and video streaming. Mobile operators are thus seeking solutions for offloading mobile data with an alternative being the use of opportunistic device-to-device communication. However, opportunistic communication has been perceived mainly as a means for disseminating delay-tolerant information. In this work we studied whether the mechanisms of opportunistic device-to-device communication could be applied to a particular class of real-time services, namely music streaming; music streaming is characterized by long-lived flows resulting into high traffic volumes. We first introduced a framework for opportunistic content sharing in the context of multimedia streaming services. We then evaluated the feasibility of opportunistic device-to-device communication for streaming data using the popular music streaming service Spotify as a use-case, and performed extensive trace-driven simulations of realistic pedestrian mobility in urban environments. The main findings of our work can be summarized as follows:

- Opportunistic device-to-device communication is viable for offloading streaming data. The performance is enhanced when the inter-download times are shorter than the mean track duration.
- The system performance depends on the node density and the request window size. In sparse areas, the system performance could be improved by adopting replay strategies to avoid buffer under-flows. In dense scenarios opportunistic device-to-device communication achieves up to 90% offload ratio even at low values of the request window size.
- The system performance exhibits low sensitivity towards the distribution of parameters such as playlist length and initial content availability. However, it is sensitive towards the mean values of these parameters. This indicates that it is sufficient to take into account mean values when designing systems that use opportunistic device-to-device communication for streaming services.

As part of our future work, we plan to implement the proposed model on actual devices and to perform field tests to verify the feasibility of our approach.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update 2014 - 2019 white paper," 2015.

[2] H. Yoon, J. Kim, F. Tan, and R. Hsieh, "On-demand video streaming in mobile opportunistic networks," in *Proc. IEEE PerCom 2008*, March 2008, pp. 80–89.

[3] V. A. Siris and D. Dimopoulos, "Multi-source mobile video streaming with proactive caching and d2d communication," in *Proc. IEEE WoWMoM*, June 2015, pp. 1–6.

[4] A. Ding, B. Han, Y. Xiao, P. Hui, A. Srinivasan, M. Kojo, and S. Tarkoma, "Enabling energy-aware collaborative mobile data offloading for smartphones," in *Proc. IEEE SECON*, June 2013, pp. 487–495.

[5] Ó. Helgason, E. A. Yavuz, S. Kouyoumdjieva, L. Pajevic, and G. Karlsson, "A mobile peer-to-peer system for opportunistic content-centric networking," in *Proc. ACM SIGCOMM MobiHeld workshop*, 2010.

[6] G. Kreitz and F. Niemela, "Spotify – large scale, low latency, p2p music-on-demand streaming," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, Aug 2010, pp. 1–10.

[7] "Spotify makes the shift to mobile with 52 percent of listening now on phones and tablets," http://techcrunch.com/2015/01/10/music-is-a-mobile-linchpin/, accessed: 2015-07-30.

[8] V. Setty, G. Kreitz, R. Vitenberg, M. van Steen, G. Urdaneta, and S. Gimåker, "The hidden pub/sub of spotify: (industry article)," in *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, ser. DEBS '13. New York, NY, USA: ACM, 2013, pp. 231–240.

[9] S. T. Kouyoumdjieva, Ó. R. Helgason, and G. Karlsson, "CRAWDAD data set kth/walkers (v. 2014-05-05)," Downloaded from http://crawdad.org/kth/walkers/, May 2014.

[10] "Legion Studio," http://www.legion.com/.

[11] Ó. Helgason, S. T. Kouyoumdjieva, and G. Karlsson, "Opportunistic communication and human mobility," *Mobile Computing, IEEE Transactions on*, vol. 13, no. 7, pp. 1597–1610, July 2014.

[12] Wi-Fi Alliance, "Wi-fi aware: Better proximity technology for personalized experiences," July 2015.

[13] Qualcomm Technologies Inc., "Lte-direct trial," Tech. Rep., Feb. 2015.

[14] S. T. Kouyoumdjieva and G. Karlsson, "Energy-aware opportunistic mobile data offloading for users in urban environments," in *IFIP Networking*, May 2015, pp. 1–9.

[15] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can wifi deliver?" *Networking, IEEE/ACM Transactions on*, vol. 21, no. 2, pp. 536–550, April 2013.

[16] L. Gao, G. Iosifidis, J. Huang, and L. Tassiulas, "Economics of mobile data offloading," in *Proc. IEEE INFOCOM*, April 2013, pp. 3303–3308.

[17] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 5, pp. 821–834, May 2012.

[18] L. Xiaofeng, H. Pan, and P. Lio, "Offloading mobile data from cellular networks through peer-to-peer wifi communication: A subscribe-and-send architecture," *Communications, China*, vol. 10, no. 6, pp. 35–46, June 2013.

[19] V. F. Mota, D. F. Macedo, Y. Ghamri-Doudanez, and J. M. S. Nogueira, "Managing the decision-making process for opportunistic mobile data offloading," in *Proc. IEEE NOMS*, May 2014, pp. 1–8.

[20] J. Morgenroth, T. Pögel, and L. Wolf, "Live-streaming in delay tolerant networks," in *Proc. ACM CHANTS*. New York, NY, USA: ACM, 2011, pp. 67–68.

[21] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: Cooperative video streaming on smartphones," in *Proc. ACM MobiSys*. New York, NY, USA: ACM, 2012, pp. 57–70.

[22] R. Jimenez, G. Kreitz, B. Knutsson, M. Isaksson, and S. Haridi, "Integrating smartphones in spotify's peer-assisted music streaming service," KTH, Royal Institute of Technology, Stockholm, Sweden, Tech. Rep. Diva-134609, 2013.

[23] J. Danihelka, D. Giustiniano, and B. Plattner, "On a cloud-controlled architecture for device-to-device content distribution," in *Proc. ACM CHANTS*, New York, NY, USA, 2015, pp. 19–24.

Analysing and Leveraging Client Heterogeneity in Swarming-based Live Streaming

Wasiur R. KhudaBukhsh*, Julius Rückert†, Julian Wulfheide†, David Hausheer†, and Heinz Koeppl*
* Bioinspired Communication Systems Lab (BCS), E-Mail: {wasiur.khudabukhsh | heinz.koeppl}@bcs.tu-darmstadt.de
† Peer-to-Peer Systems Engineering Lab (PS), E-Mail: {rueckert | julian.wulfheide | hausheer}@ps.tu-darmstadt.de,
Technische Universitaet Darmstadt, Germany

***Abstract*—Due to missing IP multicast support on an Internet scale, over-the-top media streams are delivered with the help of overlays as used by content delivery networks and their peer-to-peer (P2P) extensions. In this context, mesh/pull-based swarming plays an important role either as a pure streaming approach or in combination with tree/push mechanisms. The crucial impact of today's variety of client systems with their heterogeneous resources is not yet well understood. In this paper, we contribute to closing this gap by mathematically analysing the most basic scheduling mechanisms *latest deadline first* (LDF) and *earliest deadline first* (EDF) in a continuous time Markov chain framework and combining them into a simple, yet powerful, mixed strategy to leverage inherent differences in client resources. The contribution of this paper is, hence, twofold: (1) we develop a mathematical framework for swarming on random graphs with a focus on LDF and EDF strategies in heterogeneous scenarios; (2) we propose a mixed strategy, named SCHEDMIX, that leverages client heterogeneity. We show that SCHEDMIX outperforms LDF and EDF using different abstractions: a mean-field theoretic analysis of buffer probabilities, simulations of the stochastic model on random graphs, and a full-stack implementation of a P2P streaming system.**

I. INTRODUCTION

Media streaming dominates the traffic share on Internet. As new services are typically offered in an over-the-top (OTT) manner, they need to be efficient and scalable, without dependence on special network services. Because of its inherent limitations [3], IP multicast was not adopted in more than network islands and, in particular, is not usable for OTT content delivery. Instead, multicast functionality is realized at application layer in the form of content delivery networks (CDNs) and, to make delivery more profitable, peer-to-peer (P2P) mechanisms, or a combination of both [32]. In this work, we focus on live media streaming, an important application scenario with both high demand bandwidths and delays.

Over the years, different classes of P2P live streaming approaches were proposed [29], such as tree/push- and mesh/pull-based, as well as hybrid approaches. Due to their inherent robustness, mesh/swarming approaches continue to be of major importance, especially in hybrid settings where they often function as a substrate even when tree structures run on top of them [22], [24]. A key design issue in swarming is the data scheduling strategy used by individual peers to select chunks to be requested from their neighbours. Not only must it ensure continuous playback for an individual client, but also a healthy data replication to avoid content bottlenecks [20].

ISBN 978-3-901882-83-8 © 2016 IFIP

Several scheduling strategies of varying levels of complexity were proposed in the literature [29]. The impact of resource heterogeneity as observed in real client populations, however, is not yet fully understood. This leaves a big gap in the design space of practical P2P streaming approaches, where systematically leveraging resource imbalances could help simplifying complex scheduling strategies or designing new ones.

In this paper, we contribute to closing this gap by analysing the basic scheduling strategies *earliest deadline first* (EDF) and *latest deadline first* (LDF) based on a continuous time Markov chain framework. Our model can essentially be interpreted as a contact process [4], [15] on a random graph. An important facet of our framework is that it explicitly captures the degree-dependence of peers. Driven by the resulting analytic insights, we combine EDF and LDF into a simple, yet powerful, mixed strategy called SCHEDMIX to leverage differences in upload resources. Our theoretical efforts are complemented with a full-stack implementation of a P2P streaming system based on the SIMONSTRATOR [21] framework. The proposed strategy is shown to outperform the other two strategies using different abstractions: a mean-field theoretic analysis of buffer probabilities, simulation of the stochastic model, and discrete event-based simulation of the full-stack implementation. Thus we both theoretically and practically show the potential of using a combination of primitive scheduling mechanisms to improve overall performance in mesh-/pull-based media streaming. These results are encouraging to consider using primitive scheduling mechanism combinations in mesh-based as well as hybrid streaming and enable seamless switching (*transitions*) between them as proposed in [6].

The remainder of this paper is structured as follows: Section II presents the proposed mathematical framework, followed by our mean-field analysis in Section III. Subsequently, Section IV presents our findings from simulations of the stochastic process. Section V is devoted to the full-stack implementation. Finally, in Section VI we discuss related work and conclude our paper with a discussion in Section VII.

II. MODEL

A. The network

We describe the underlying network as a random graph. We assume the associated degree distribution has a finite mean. Let

$\mathcal{G}_M$ be the class of all simple and connected random graphs with M nodes. Let $\pi : \mathbb{N} \to [0,1]$ be the associated degree distribution. We also define the size-biased degree distribution, q as follows:

$$q(k) := \frac{k\pi(k)}{\sum_k k\pi(k)}, \tag{1}$$

for $k \in \mathbb{N}$. The quantity $q(k)$ is the probability that a given edge points to a vertex of degree k.

B. The peer-to-peer communication system

Suppose there are M peers and a single server. Let n denote the buffer length. The server uniformly selects a peer at random and uploads a chunk at buffer position 1. It continues to upload chunks to the chosen peer until there is a connection breakage/loss (an event that occurs with a small probability, say $\varepsilon \in (0,1]$) in which case the server chooses a peer again uniformly at random. The chunk at buffer position n, if available, is pushed for playback. After playback, the chunk is removed and all other chunks are shifted one index closer to playback. Each peer maintains a Poisson clock with rate proportional to its degree[1]. A peer, if not selected by the server, contacts one of its neighbours uniformly at random at each tick of its Poisson clock and seeks to download a missing chunk. The chunk it downloads from among all downloadable chunks is decided by its chunk selection strategy. For simplicity, we assume that the playback rate is one chunk per unit of time.

Let $\mathcal{G} := (\mathcal{V}, \mathcal{E}) \in \mathcal{G}_M$ be a given realisation of a random graph, where $\mathcal{V}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the sets of vertices and edges, respectively. Each node is a peer. Let $\Omega := \{\omega \in \{0,1\}^{M\times n} \mid \sum_{i=1}^{M} \omega(i,1) = 1\}$ be the configuration space of all peers and buffers, and denote all subsets of Ω by $\mathcal{A}$. Define a continuous time Markov chain (CTMC) $\{X_t\}_{t\geq 0}$ on the measurable space $(\Omega, \mathcal{A})$ as $X_t(i,j) := 1$ if the j-th buffer location of the i-th peer is filled, and 0 otherwise. The rows of the matrix X_t, denoted as $X_t^1, X_t^2, \ldots, X_t^M$ represent buffer states of peers $1, 2, \ldots, M$, respectively.

Let $\mathbb{S} : \{0,1\}^{M\times n} \cup \{0,1\}^n \to \{0,1\}^{M\times n} \cup \{0,1\}^n$ denote the buffer shifting operator defined as $\mathbb{S}Y := (0, y_1, y_2, \ldots, y_{n-1})$ for $Y = (y_1, y_2, \ldots, y_n) \in \{0,1\}^{M\times n} \cup \{0,1\}^n$ where $y_1, y_2, \ldots, y_n$ denote the columns of Y.

Let us now define the transition rates of interaction for a node $v \in \mathcal{V}$ as follows

$$\mu^v(u, u+e_i) = \begin{cases} \displaystyle\sum_{l\in\mathcal{V}:(v,l)\in\mathcal{E}} \varsigma \mathbb{1}(X_t(l,i)=1)\alpha^v(i,u,X_t^l), & \\ \qquad\qquad\qquad\qquad \text{if } i \neq 1, & (2)\\ \mathbb{1}(X_t(v,1)=1)(1-\varepsilon+\varepsilon/M) & \\ \quad +\mathbb{1}(X_t(v,1)=0)\varepsilon/M \qquad \text{if } i = 1, & \end{cases}$$

where $u = (u_1, u_2, \ldots, u_n) \in \mathcal{T} := \{0,1\}^n$, $i \in \mathcal{F} := \{1, 2, \ldots, n\}$, such that $u_i = 0$, $\varsigma > 0$ is a constant, $\mathbb{1}(.)$ is the indicator function, e_i is the i-th unit basis vector of the n-dimensional Euclidean space and $\alpha^v : \mathcal{F} \times \mathcal{T} \times \mathcal{T} \to [0,1]$ is the chunk selection function of the peer $v \in \mathcal{V}$. In words, $\alpha^v(i,u,X_t^l)\delta t$ is the probability of downloading chunk i when peer v is in buffer state u and contacts peer l in buffer state X_t^l. We defer an elaborate discussion of the chunk selection function to a later section. The system is described by the following *master equation*

$$\begin{aligned}\frac{d\mathrm{P}(X)}{dt} = &-\mathrm{P}(X) + \sum_{v'\in\mathcal{V}} \mathbb{1}(X(v',1)=1)\Big[\sum_{Y\in\Omega:\mathbb{S}Y=X-\Delta(v',1)} \\ &\mu^{v'}(Y^{v'}, Y^{v'}+e_1)\Big\{\mathrm{P}(Y) \\ &+ \sum_{i\in\mathcal{F}\setminus\{1\}}\sum_{v\in\mathcal{V}\setminus\{v'\}}\Big(\sum_{Z\in\Omega:Z=Y-\Delta(v,i)} \mu^v(Y^v-e_i, Y^v) \\ &\times \mathrm{P}(Z) - \mu^v(Y^v, Y^v+e_i)\mathrm{P}(Y)\Big)\Big\}\Big],\end{aligned} \tag{3}$$

for $X \in \Omega$, where $\Delta(v,i)$ is an $M \times n$ matrix of all zeroes except for a unity at position (v,i). We omit the time index whenever dependence is unambiguous.

The master equation (3) can not be solved analytically. We, therefore, carry out an aggregation of the chain into population counts. Define $\deg(v) := \sum_{l\in\mathcal{V}} \mathbb{1}((v,l) \in \mathcal{E})\ \forall v \in \mathcal{V}$ and $\mathcal{D} := \{d \mid \exists v \in \mathcal{V}, \deg(v) = d\}$. Consider a map T defined by $\mathrm{T}(X) := (z_x^k : x \in \mathcal{T}, k \in \mathcal{D})$ where $z_x^k := \sum_{v\in\mathcal{V}} \mathbb{1}(X^v = x)\mathbb{1}(\deg(v) = k)$, the number of degree-$k$ peers at buffer configuration x. Define an equivalence relation $\overset{\mathrm{T}}{\sim}$ on Ω as $X \overset{\mathrm{T}}{\sim} Y \iff \mathrm{T}(X) = \mathrm{T}(Y)$ and $\Omega_t := \{X \in \Omega : \mathrm{T}(X) = t\}$ for each t. Then, $\{\Omega_t\}$ is a partition of Ω and each Ω_t is an equivalence class. The induced probability is given by

$$\mathrm{P}(\mathrm{T}(X) = t) = \sum_{X\in\Omega:\mathrm{T}(X)=t} \mathrm{P}(X). \tag{4}$$

Such an aggregation is useful in reducing the state space if we now consider the lumped process T of population counts instead. In [10], we provide a necessary and sufficient condition for such an aggregation to engender state space reduction and also discuss worst case scenarios. We emphasize that we do lose information in the process of aggregation. Also, the lumped process is not necessarily Markovian [9].

III. Mean-field theoretic analysis

In this section, we approximate the lumped process T defined in Section II-B, when M is large. Mean-field theory is extensively used for this purpose [4], [13], [18]. As a first step in this direction, peers are assumed to be independently interacting with a mean environment. This allows us to treat each neighbour of a degree-k peer as an independent sample from a mean environment. We also impose that peers having the same degree play the same chunk selection strategy and thus, behave indistinguishably in a large random graph, suggesting that such a mean-field behaviour can very well be described by population counts. We, therefore, define a mean-field population model that lumps the original process according to the equivalence relation $\overset{\mathrm{T}}{\sim}$. We shall index all the relevant quantities by degree k in the following, instead of indexing by peers.

[1] That is, we place a Poisson clock on each edge of the graph.

A. Mean-field master equations

Consider the process $\{Z_t\}_{t\geq 0}$ defined as $Z_t \coloneqq (z_x^k(t) : x \in \mathcal{T}, k \in \mathbb{N})$ where $z_x^k(t)$ is the number of degree-k peers at buffer configuration $x \in \mathcal{T}$ at time t. We get our mean-field transition rates for a degree-k peer as follows, for each $k \in \mathbb{N}, u \in \mathcal{T}$ and $i \in \mathcal{F} \setminus \{1\}$ such that $u_i = 0$,

$$\begin{aligned}\beta^k(u, u+e_i) &= \sum_{l=1}^{k} \varsigma \mathrm{E}[\mathbb{1}(Y_l(i)=1)\alpha^k(i,u,Y_l)] \\ &= k\varsigma \mathrm{E}[\mathbb{1}(Y_1(i)=1)\alpha^k(i,u,Y_1)],\end{aligned}$$

where $\{(Y_l, d_l) \mid Y_l = (Y_l(1), Y_l(2), \ldots, Y_l(n)) \in \mathcal{T}, d_l \in \mathbb{N}\}_{l=1}^k$ is a set of k independent and identically distributed (i.i.d.) samples from the mean environment of a degree-k peer. The first component of each neighbour is the buffer state and the second component, its degree. Note that d_l's are distributed according to q of eq. (1). Then,

$$\begin{aligned}&\mathrm{E}[\mathbb{1}(Y_1(i)=1)\alpha^k(i,u,Y_1)] \\ &= \sum_{v\in\mathcal{T}:v_i=1} \sum_{m\in\mathbb{N}} \alpha^k(i,u,v)\mathrm{P}(Y_1 = v \mid d_1 = m)\mathrm{P}(d_1 = m) \\ &= \sum_{v\in\mathcal{T}:v_i=1} \sum_{m\in\mathbb{N}} q(m)\frac{\mathrm{E}[z_v^m]}{n_m}\alpha^k(i,u,v).\end{aligned}$$

where n_m is the number of peers of degree m. Thus, we get,

$$\beta^k(u, u+e_i) = k\varsigma \sum_{v\in\mathcal{T}:v_i=1} \sum_{m\in\mathbb{N}} q(m)\frac{\mathrm{E}[z_v^m]}{n_m}\alpha^k(i,u,v), \quad (5)$$

for each $k \in \mathbb{N}$, $u \in \mathcal{T}$ and $i \in \mathcal{F} \setminus \{1\}$ such that $u_i = 0$. For $i = 1$, we set β such that $\sum_{u\in\mathcal{T}:u_1=1} \frac{z_{u-e_1}^k}{n_k}\beta^k(u-e_1, u) = \frac{1}{M}$, the total input to the system by the server. Define the change vector $\varrho : \mathbb{N}\times\mathcal{T}\times\mathcal{F} \to \{-1,0,1\}^{|\mathcal{T}|\times\mathbb{N}}$ such that $Y = Z - \varrho(k,u,i) \implies y_u^k = z_u^k + 1, y_{u+e_i}^k = z_{u+e_i}^k - 1, y_x^l = z_x^l \forall l \in \mathbb{N}\setminus\{k\}, x \in \mathcal{T}\setminus\{u\}$. Broadening the scope of definition of β by setting it to 0 for all $u, u+e_i$ not covered in eq. (5), for large M, we have the following *mean-field master equation*

$$\begin{aligned}\frac{d\mathrm{P}(Z)}{dt} = &-\mathrm{P}(Z) + \sum_{\substack{Y:\sum_{\mathbb{S}v=u} y_v^l = z_u^l \\ \forall u,v\in\mathcal{T}, l\in\mathbb{N}}} \Bigg[\mathrm{P}(Y) \\ &+ \sum_{l\in\mathbb{N},u\in\mathcal{T},i\in\mathcal{F}} (y_u^l + 1)\beta^l(u, u+e_i) \\ &\times \mathrm{P}(Y - \varrho(l,u,i)) \\ &- \sum_{l\in\mathbb{N},u\in\mathcal{T},i\in\mathcal{F}} y_u^l\beta^l(u,u+e_i)\mathrm{P}(Y)\Bigg].\end{aligned} \quad (6)$$

In pursuance of the mean dynamics, we begin by first setting $\mathrm{P}(Y) = 0\,\forall\, Y \notin \mathbb{N}_0^{|\mathcal{T}|\times\mathbb{N}}$ where $\mathbb{N}_0 \coloneqq \mathbb{N}\cup\{0\}$, and then by defining, for each $l \in \mathbb{N}, u \in \mathcal{T}, i \in \mathcal{F}$, the following quantity $\gamma_{l,u,i}(Z) \coloneqq z_u^l\beta^l(u, u+e_i)$. Next, we note that, in mean field, we can write $\mathrm{E}[\gamma_{l,u,i}(Z)]$ as $\mathrm{E}[z_u^l]\beta^l(u,u+e_i)$. The following result encapsulates the mean dynamics of the system.

Result 1. *The process $\{Z_t\}_{t\geq 0}$ admitting master equation (6) satisfies*

$$\frac{d\mathrm{E}[Z]}{dt} = -\mathrm{E}[Z] + \mathrm{E}[Y] + \sum_{l\in\mathbb{N},u\in\mathcal{T},i\in\mathcal{F}} \varrho(l,u,i)\mathrm{E}[\gamma_{l,u,i}(Y)], \quad (7)$$

where $Y \in \mathbb{N}_0^{|\mathcal{T}|\times\mathbb{N}}$ is such that $y_u^l = \sum_{\mathbb{S}v=u} z_v^l \forall l \in \mathbb{N}, u \in \mathcal{T}$.

The proof is provided in [10]. Looking closely at eq. (7) and recalling the definition of $\varrho(l,u,i)$, we write down explicitly, for each $u \in \mathcal{T}, k \in \mathbb{N}$

$$\begin{aligned}\frac{d\mathrm{E}[z_u^k]}{dt} = &-\mathrm{E}[z_u^k] + \sum_{v\in\mathcal{T}:\mathbb{S}v=u} \Bigg[\mathrm{E}[z_v^k] \\ &+ \sum_{i\in\mathcal{F}} \mathrm{E}[z_{v-e_i}^k]\beta^l(v-e_i, v) - \sum_{i\in\mathcal{F}} \mathrm{E}[z_v^k]\beta^l(v, v+e_i)\Bigg],\end{aligned} \quad (8)$$

a self-consistent (autonomous) set of ordinary differential equations (ODEs) for the mean population counts.

It is convenient to work with proportions to study the mean dynamics. Therefore, define $W_t \coloneqq (w_x^k(t) : x \in \mathcal{T}, k \in \mathbb{N})$ where $w_x^k(t) \coloneqq z_x^k/n_k$. We argue that, when the number of peers is large, it suffices to study the mean dynamics of the proportions, for the fluctuation around mean is expected to be negligible for large systems [12]. Therefore, denoting $\mathrm{E}[w_x^k]$, with abuse of notation, by w_x^k itself, we write down the following *rate equations*,

$$\begin{aligned}\frac{dw_u^k}{dt} = &-w_u^k + \sum_{v\in\mathcal{T}:\mathbb{S}v=u} \Bigg[w_v^k \\ &+ \sum_{i\in\mathcal{F}} \Big(w_{v-e_i}^k\beta^k(v-e_i, v) - w_v^k\beta^k(v, v+e_i)\Big)\Bigg],\end{aligned} \quad (9)$$

for each $u \in \mathcal{T}, k \in \mathbb{N}$. We find steady-state proportions by setting $\frac{dw_u^{(k)}}{dt} = 0$, giving rise to following fixed point equations at steady state,

$$\begin{aligned}w_u^k = \sum_{v\in\mathcal{T}:\mathbb{S}v=u} \Bigg[w_v^k + \sum_{i\in\mathcal{F}} \Big(&w_{v-e_i}^k\beta^k(v-e_i, v) \\ &- w_v^k\beta^k(v, v+e_i)\Big)\Bigg].\end{aligned} \quad (10)$$

Observe that $\sum_{u\in\mathcal{T}} \frac{dw_u^k}{dt} = 0$ for all $k \in \mathbb{N}$. This is because of the fact that proportions sum up to 1, *i.e.*, $\sum_{u\in\mathcal{T}} w_u^k = 1 \forall k \in \mathbb{N}$. It does merit some attention that the population model presented here can be thought of as an infection model with 2^n distinct levels of a disease, each level being represented by a $u \in \mathcal{T}$ and (gradual) recovery being represented by the shifting of buffer state after playback. This amounts to saying, a peer with all buffer positions filled is infected to the highest extent of a disease and if it does not download any chunk, *i.e.*, if it does not get infected, within the next n-time units, it will gradually recover to a state of complete susceptibility (no chunk available).

One of the key metrics of performance in live streaming context is the buffer probability. The buffer probability of

index i of a degree-k peer is the probability that a degree-k peer has a chunk at buffer index i. In mean field, this becomes the proportion of degree-k peers that have chunks at buffer index i. Therefore, we define $p_k : \{1, 2, \ldots, n\} \to [0, 1]$, the buffer probability of a peer of degree $k \in \mathbb{N}$ as

$$p_k(i) = \sum_{u \in \mathcal{T} : u_i = 1} w_u^k. \tag{11}$$

The corresponding global performance of the network is linked to these degree-specific buffer probabilities through the associated degree distribution of $\mathcal{G}$ as follows

$$p(i) = \sum_{k \in \mathbb{N}} \pi(k) p_k(i). \tag{12}$$

Next, we try to derive a recurrence relation among p_k's by means of eq. (10) to understand their behaviour. We have the following result in that direction.

Result 2. *The process $\{W_t\}_{t \geq 0}$ of proportions obeying rate equation eq.* (9)*, admits the following recursion relation among the buffer probabilities at steady state*

$$p_k(i+1) = p_k(i) + \sum_{u \in \mathcal{T} : u_i = 1} w_{u-e_i}^k \beta^k(u - e_i, u)$$
$$p(i+1) \;\; = p(i) + \sum_{k \in \mathbb{N}} \pi(k) \sum_{u \in \mathcal{T} : u_i = 1} w_{u-e_i}^k \beta^k(u - e_i, u)$$

for all $i, k \in \mathbb{N}$. Moreover, buffer probabilities are nondecreasing functions of their arguments, i.e., buffer indices.

The proof is omitted for want of space and is given in [10].

Interpretation of result 2: The left hand side of the recurrence relation gives the probability that the chunk required to fill the buffer location $i + 1$ is present. The right hand side tells us that there are two possible ways to have the chunk at buffer index $i + 1$ present. First, it could already be there at buffer index i, with probability of buffer index i, and was made available at index $i + 1$ due to shifting. Second, the chunk was not there, but the peer could download it in the mean time. Roughly speaking, this occurs with probability $\sum_{u \in \mathcal{T} : u_i = 1} w_{u-e_i}^k \beta^k(u - e_i, u)$ for a degree-k peer. This forms the basis of our further analysis of buffer probabilities.

Now we make use of a largely adopted assumption about the chunk selection function. We assume that the chunk selection function of a degree-k peer, $\alpha^k(i, u, v)$ does not depend on any particular value of u and v, but rather assigns probability to buffer indices according to their relative importance as pronounced by EDF and LDF. Call this simplified policy s_k, instead of α^k. This implies,

$$\begin{aligned}\beta^k(u, u + e_i) &= k\varsigma \sum_{v \in \mathcal{T} : v_i = 1} \sum_{l \in \mathbb{N}} q(l) w_v^l \alpha^k(i, u, v) \\ &= k\varsigma s_k(i) \sum_{l \in \mathbb{N}} q(l) p_l(i) = k\varsigma s_k(i) \theta_i,\end{aligned}$$

where $i \in \mathcal{F}$ and $\theta_i := \sum_{l \in \mathbb{N}} q(l) p_l(i)$ encapsulates the probability that an arbitrarily given edge points to a node where chunk i is available.

Let us now revisit the recurrence relation in result 2 and plug in the above simplified quantities. In order to do so, note that, for all $i \in \mathcal{F}$,

$$\begin{aligned}\sum_{u \in \mathcal{T} : u_i = 1} w_{u-e_i}^k \beta^k(u - e_i, u) &= \sum_{v \in \mathcal{T} : v_i = 0} w_v^k \beta^k(v, v + e_i) \\ &= k\varsigma\theta_i s_k(i) \sum_{v \in \mathcal{T} : v_i = 0} w_v^k \\ &= k\varsigma\theta_i (1 - p_k(i)) s_k(i).\end{aligned}$$

The recursion relation in result 2 then reads

$$p_k(i+1) = p_k(i) + k\varsigma\theta_i (1 - p_k(i)) s_k(i), \tag{13}$$

where $k \in \mathbb{N}$, $i = 1, 2, \ldots, n-1$, and $\varphi := p_k(1) = \frac{1}{M}$. Such a recurrence relation in the special case of a homogeneous system has served as a starting point for the study of buffer probabilities in a number of articles in the literature, *e.g.*, [27], [33], [34]. In fact, by choosing $\pi(k) = \mathbb{1}(k = k^*), \varsigma = \frac{1}{k^*}$ for some $k^* \in \mathbb{N}$, we retrieve from eq. (13) the corresponding recurrence relation in the homogeneous setup, as found in [27], [33], [34]. Our endeavour was to provide a principled approach to derive such a recurrence relation in a more general heterogeneous setup exhibiting degree dependence of peers.

Remark. Equations (12) and (13) are two key instruments in our analysis of buffer probabilities. While eq. (13) describes the playback experience of a degree-k peer, a local aspect, eq. (12) allows us to combine these local information through degree distributions of arbitrary networks to give us a global view. This is notable because even this simple, approximate model allows us to capture the dependence of performance on network structure by plugging in its degree distribution.

We shall now focus on the two popular chunk selection strategies, namely, LDF and EDF. We follow the same interpretations of EDF and LDF as laid down in [34].

B. Chunk selection function

1) Latest deadline first (LDF) strategy: This strategy aims to download the rarest piece first. The priority is thus on the initial buffer indices. Therefore, $s_k(i)$ can be written as

$$s_k(i) = [1 - \varphi] \prod_{j=1}^{i-1} \left[p_k(j) + (1 - p_k(j))(1 - k\varsigma\theta_j) \right].$$

The explanation, omitted for want of space, is simple and is provided in [10]. This gives us the following result.

Result 3. 1) *The chunk selection function for the latest deadline first (LDF) strategy can be expressed as*

$$s_k(i) = 1 - p_k(i). \tag{14}$$

2) *The recursion relation for buffer probabilities for the latest deadline first (LDF) strategy has the following form, for $i = 1, 2, \ldots, n-1$ and $k \in \mathbb{N}$*

$$p_k(i+1) = p_k(i) + k\varsigma\theta_i (1 - p_k(i))^2. \tag{15}$$

The proof is similar to [34], however, for the sake of completeness, it is provided in [10].

2) Greedy strategy: The greedy strategy or the earliest deadline first (EDF) strategy seeks to download pieces that are close to playback. The priority is thus on playback urgency and hence on the final buffer indices. Therefore, the chunk selection function can be expressed as

$$s_k(i) = [1-\varphi] \prod_{j=i+1}^{n-1} \left[p_k(j) + (1-p_k(j))(1-k\varsigma\theta_j)\right].$$

The explanation is similar to the case of the LDF strategy, with the notable exception that now we require to search buffer index n first, then $n-1$ and so on.

Result 4. 1) *The chunk selection function for the greedy strategy (EDF) can be expressed as*

$$s_k(i) = 1 - \varphi - p_k(n) + p_k(i+1). \tag{16}$$

2) *The recursion relation for buffer probabilities for the greedy strategy (EDF) has the following form, for* $i = 1, 2, \ldots, n-1$ *and* $k \in \mathbb{N}$

$$\begin{aligned} p_k(i+1) = p_k(i) + k\varsigma\theta_i(1-p_k(i)) \\ [1-\varphi-p_k(n)+p_k(i+1)]. \end{aligned} \tag{17}$$

The proof is provided in [10].

Remark. A typical EDF buffer probability curve exhibits a late, sharp increase, contrary to an LDF curve (see [33], [34]). However, when M is large, EDF hinders propagation of new chunks. While LDF is known to possess good scalability, EDF outperforms LDF when M is small. We wish to exploit this feature of EDF even when M is large. In order to do so, we must devise a way to arrest this content bottleneck. We conjecture that this can be done by employing a reasonably small percentage of strong peers (the ones with higher bandwidth, say, but not necessarily connected directly to the server) to play LDF so as to act as *pseudo-servers* in the system. We pursue this idea by studying different strategy profiles in a minimal setup with only two degrees, where we call the peers of higher degree strong peers and peers of smaller degree, weak peers.

Suppose there are only two degrees $k_1, k_2 \in \mathbb{N}$ in the system where $k_1 < k_2$. For typographical convenience, we shall subscript all the relevant variables with only $1, 2$ instead of k_1, k_2 respectively, whenever the degree of a vertex appears as a subscript or as an argument to a function, *e.g.*, π_1, π_2 in place of $\pi(k_1), \pi(k_2)$ respectively and $p_1(i), p_2(i)$ in place of $p_{k_1}(i), p_{k_2}(i)$ respectively.

C. Pure LDF strategy

As seen in Section III-B1, buffer probabilities for the two degrees k_1, k_2 when everybody plays LDF, are given by the following recursion relations

$$\begin{aligned} p_1(i+1) = \quad & p_1(i) + k_1\varsigma\theta_i(1-p_1(i))^2, \\ p_2(i+1) = \quad & p_2(i) + k_2\varsigma\theta_i(1-p_2(i))^2, \end{aligned} \tag{18}$$

for $i = 1, 2, \ldots, n-1$. We adopt a continuous approximation of the above two difference equations (as done in [27], [34], for instance). Treating the buffer index i as a continuous variable x and writing y_1, y_2, θ for $p_1(i), p_2(i)$ and θ_i respectively, we have the following differential equations

$$\begin{aligned} \frac{dy_1}{dx} &= k_1\varsigma\theta(1-y_1)^2, \\ \frac{dy_2}{dx} &= k_2\varsigma\theta(1-y_2)^2. \end{aligned} \tag{19}$$

The above luckily allows an exact solution which we present in the next result.

Result 5. *For the pure LDF strategy and large systems, i.e., when* $M \to \infty$*, the two buffer probabilities are related according to the following equation*

$$y_2 = \frac{y_1}{r + (1-r)y_1}, \tag{20}$$

where $r = \frac{k_1}{k_2}$ *is the relative strength of the weak peers compared to the strong ones.*

The proof is given in [10]. We immediately see that $y_2 > y_1$, *i.e.*, the stronger peers have better performance owing to their greater rate of interaction. However, this difference in performance for the weak peers due to degree disparity can be made arbitrarily small if a sufficiently large buffer is made available. Another interesting consequence is that the above can now be used to derive an expression for buffer-size requirements and facilitate sensitivity analysis therefrom. That is, given $\epsilon_1 = 1-p_1(n)$, the playback discontinuity of the weak peers, we can find the required buffer length of the weak peers $n_1 = f(\pi, r, \epsilon_1)$ that ensures performance at level ϵ_1 for some f[2]. Notice that the global performance is related to ϵ_1 by

$$1 - \epsilon = \pi_1(1-\epsilon_1) + \pi_2 \frac{1-\epsilon_1}{1-(1-r)\epsilon_1},$$

where $1 - \epsilon = p(n)$. This can be used when we intend to achieve a prespecified level of global performance.

D. Mixed strategy: SchedMix

Now we turn to the mixed strategy referred to as SchedMix. Suppose the weaker peers of degree k_1 adopt EDF and the stronger peers of degree k_2, LDF. Following Sections III-B1 and III-B2, we have the following recursion relations

$$\begin{aligned} p_1(i+1) &= p_1(i) + k_1\varsigma\theta_i(1-p_1(i)) \\ &\qquad [1-\varphi-p_1(n)+p_1(i+1)], \\ p_2(i+1) &= p_2(i) + k_2\varsigma\theta_i(1-p_2(i))^2, \end{aligned} \tag{21}$$

for $i = 1, 2, \ldots, n-1$. As before, we shall use a continuous approximation to study their behaviour. Writing $\epsilon_1 = 1-p_1(n)$, we get the following differential equations:

$$\begin{aligned} \frac{dy_1}{dx} &= \frac{k_1\varsigma\theta(1-y_1)(y_1-\varphi+\epsilon_1)}{1-k_1\varsigma\theta(1-y_1)}, \\ \frac{dy_2}{dx} &= k_2\varsigma\theta(1-y_2)^2. \end{aligned} \tag{22}$$

[2]The exact expression is provided in [10].

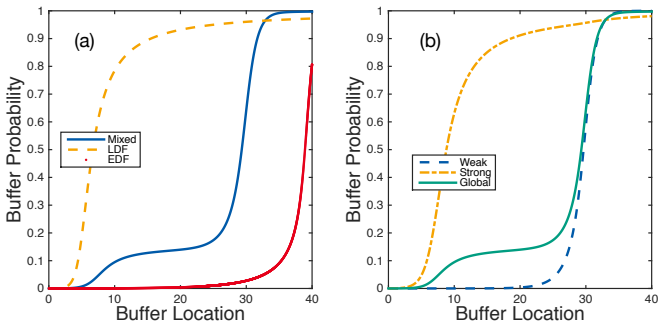


Fig. 1: Performance comparison based on mean-field analysis of buffer probabilities. (a) Global buffer probabilities for the three strategy profiles. SCHEDMIX gives higher playback continuity than both EDF and LDF for the given buffer length. (b) Comparison of weak versus strong under SCHEDMIX. Weak peers indeed eventually outperform the strong peers under SCHEDMIX. Parameter values: $M = 10000, k_1 = 5, k_2 = 15, \pi_1 = 0.85 = 1 - \pi_2, \varsigma = 0.20$.

The above equations, unfortunately, do not yield an analytic solution. Therefore, we resort to numerical solution to compare global performance of the system under different strategy profiles. It turns out that performance under SCHEDMIX is indeed better than that under the pure LDF strategy (see Fig. 1), substantiating our claim.

When we compared performance of weak peers versus strong ones, an interesting phenomenon was observed. The weak peers could eventually manage to outperform the strong ones, caused by a sharp increase in buffer probabilities that a typical "EDF curve" enjoys and what we call the boon of heterogeneity (see Fig. 1). This phenomenon is in agreement with our supposition and can be explained intuitively. Both strong and weak peers benefit from being exposed to a heterogeneous environment. In a homogeneous setup, one would expect somewhat similar availability of chunks among all its neighbours. On the contrary, a heterogeneous environment makes available a diverse collection of chunks. This prepones the steep rise that a typical "EDF curve" enjoys. Since an EDF curve has a greater growth-rate in the neighbourhood of 1 (see [33], [34]), weak peers can eventually outperform LDF-playing strong peers even for moderate buffer-lengths.

Remark. We do not consider the pure EDF strategy separately here as it can be studied in a similar fashion. In [10], we also provide a short stability analysis that gives an additional justification of why the weak peers outperform the strong ones.

IV. SIMULATION OF THE STOCHASTIC MODEL

In this section, we document our findings from the simulation of the stochastic model. This is carried out in two steps: first, generation of a random graph and second, simulation of the content delivery process in accordance with Section II.

We dispense with a description of how to simulate CTMCs due to insufficiency of space. Interested readers are referred to [10] where we also investigated the effect of assuming an exponential shifting time versus a deterministic one and confirmed that the behaviour of the strategies remained unaffected.

Startup latency: The second metric that we look at is the start-up latency. It is the time a peer should wait before starting playback. While there is no unanimity as to how one should define this quantity, it is reasonable to wait until a newly arrived peer's buffer attains a steady state. If it starts playback before that, it is likely to experience below steady state playback quality initially. On the other hand, waiting longer will not improve long-term playback experience. In a homogeneous set-up where everybody plays the same policy and has the same buffer probabilities, as argued in [33], this is well represented by $\sum_i p(i)$, the average number of available chunks at each peer. In our heterogeneous model, a higher degree peer interacts more often than a lower degree peer. Therefore, a newly arrived degree-k peer should have start-up latency of $k\varsigma \sum_i p(i)$ in the mean-field. The corresponding global metric follows as $\mathrm{E}[k]\varsigma \sum_i p(i)$. For aesthetic reasons, we normalise this quantity to $(0, 1)$.

Impact of network structure: In order to see the impact of network structure, we perform simulation of the model on Barabási-Albert (BA) preferential attachment [2] and Watts-Strogatz (WS) small world [25] networks. Simulation results on a BA network with 2000 peers (with 25% of them playing LDF) and that on a WS network with 5000 peers (with 20% of them playing LDF) are depicted in Fig. 2. In both cases, the mixed strategy SCHEDMIX gives a better performance, corroborating our claim. More importantly, it causes a significant reduction in start-up latency.

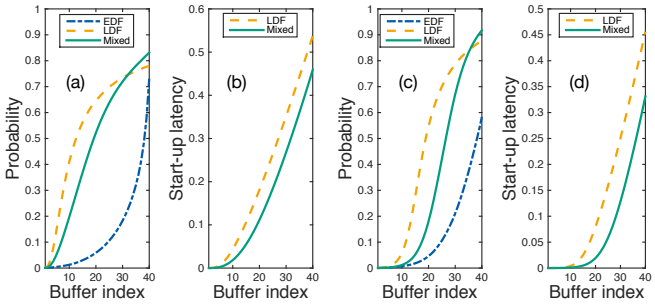


Fig. 2: Impact of network structure and performance evaluation in terms of buffer probabilities and the start-up latency on a Barabási-Albert (BA) and a Watts-Strogatz (WS) graph. Figures (a), (b) show performance on a BA graph with 2000 peers. Figures (c), (d) display performance on a WS graph with 5000 peers. In both cases, $n = 40, \varsigma = 0.25$. Please note that start-up latency is shown only for strategies ensuring playback continuity of at least 0.75 with buffer size $n = 40$.

Remark. Although Fig. 2 stands affirmatory to the fact that SCHEDMIX does outperform the pure LDF and the pure EDF strategies, the crux of employing SCHEDMIX remains in letting most peers play greedy. SCHEDMIX, thus, allow for

smaller start-up latency to ensure good playback performance for everyone (at least as good as pure LDF strategy). This is a significant benefit.

V. Full-stack simulation study

A. Practical system model

We also designed and implemented a practical P2P live video streaming system and considered communication network-related factors. The implementation is based on the SIMONSTRATOR API [21] and is evaluated using the network simulation framework PEERFACTSIM.KOM [23]. The full-stack implementation includes protocols for mesh establishment and maintenance, the scheduling mechanisms themselves, and buffer management. Due to space constraints, we omit some details that follow state-of-the-art P2P streaming systems and in particular [22], [26].

1) Mesh establishment: For the establishment of the mesh overlay structure, a join procedure is implemented that uses a BitTorrent-like tracker as central node registry. The tracker selects a maximum of 30 neighbours uniformly at random from the set of currently active peers and sends the list to the requesting peer. For both in- and outgoing directions, peers calculate a maximum number of connections by dividing 90% of the available bandwidth by the video bitrate and rounding the result to the next integer. Joining peers strive to fill their free incoming connections and thus contact multiple peers from the initial neighbour list in parallel and query the tracker for additional contacts if necessary. Receivers accept the requests depending on the availability of free connection slots. Limiting the number of connections is combined with a per-connection transfer queue to avoid too many parallel transfers that could stall each other, leading to a situation where video chunks would take an indefinite time to be delivered. In case a peer has no free connection slots, requests can still be accepted with a small probability to foster randomness in the mesh structure where early peers might otherwise be already blocked, and to allow peers with high bandwidths to eventually become well connected. The second aspect turned out to be important for applying the proposed mixed scheduling strategy, which relies on degree heterogeneity across peers.

2) Scheduling and data exchange: The actual scheduling of data transmissions is done by each peer individually based on its local clock with a rate proportional to its in-degree and buffer status. Chunks are selected by the scheduling strategy from a defined request window on the local buffer, which is used to limit the chunks requested. For the simulations, the window size is set to a default value of 20, starting at the beginning of the buffer for LDF and the end for EDF. The selected chunks are assigned uniformly at random to a peer's in-connections, are batched on a per-neighbour basis into chunk requests, and are sent out. The buffer length is 4 seconds, translating to 50 chunks at a rate of 8 chunks/s.

3) Playback policy: A simplified policy was realized for this initial simulation study. Joining peers learn about the current broadcasting position from the tracker and start their playback after 4 seconds (the buffer length). In the meantime, they establish connections and start requesting chunks. Once started, the playback proceeds based on the local clock and at the video bitrate. Chunks that miss the playback deadline are recorded in terms of *playback continuity*.

B. Full-stack simulation results

A simulative sensitivity analysis was conducted, covering key system and environment parameters. Due to space constraints, only results for the default configuration are presented here (see [10] for additional results). The defaults were obtained by conducting calibration runs for several parameter combinations. Due to the large configuration space, ensuring an overall optimal configuration was not possible and is hard to achieve in general. The presented results, thus, focus on studying the potential of the proposed scheduling mechanism, not on the absolute performance. All simulations were repeated 30 times with different random seeds. 95%-confidence intervals are reported for all mean values.

Simulation workload: Peers are divided into three resource classes based on bandwidth distributions reported in [17] (see Table I). We acknowledge that these bandwidths are rather high in comparison to configurations used in related works. Yet, we intend to reflect a setting in that the delivery is not primarily limited by peer bandwidths but rather focus on content bottlenecks resulting from the scheduling strategy itself [7]. This is important as peers can only use available bandwidth if scheduling ensures a timely replication of chunks. At the beginning of the simulation scenario, peers subsequently join the system in a random order and at a constant arrival rate. After the system stabilizes, performance and cost metrics are recorded on per-peer basis and aggregated for 60 seconds intervals and over the total simulation time of 90 minutes. Peers stay in the system until the end of the simulation.

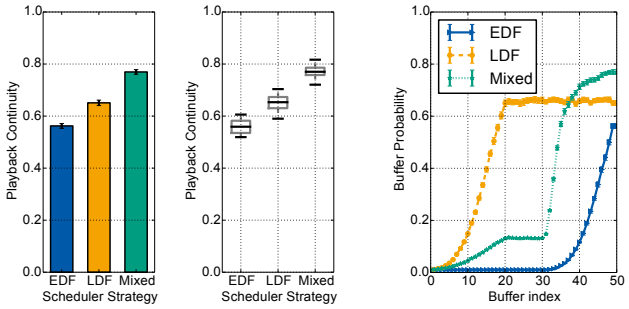
TABLE I: Used peer bandwidth distribution based on [17].

Class	Number	Share	UL BW (Mbps)	DL BW (Mbps)
Low	50	50%	5	26
Medium	30	30%	4.5	60
High	20	20%	56	134

As observed in [14], the source bandwidth plays an important role and, thus, its influence is studied in more detail in [10]. Per default, a single source is used with an up-/download bandwidth of 12.5Mbps. With a video bitrate of 1,500 Kbps as commonly observed and recently reported [11], this translates to a maximum out-degree of 7.

Figure 3 shows the streaming performance for the default configuration. Here, it is to note that using a request window of size 20 is an extreme case as it artificially limits the request rate by localizing the chunks to be selected. This is done to highlight the key difference between the chunk selection strategies. Other configurations of this parameter are presented in [10]. Figure 3a shows that SCHEDMIX achieves a significantly higher playback continuity compared to the pure strategies. The buffer probability (see Figure 3b) shows that

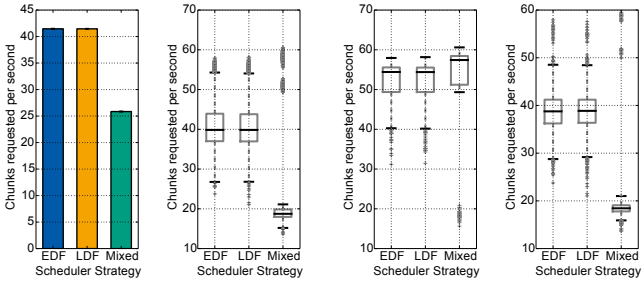
an early replication of new chunks greatly supports the greedy replication by EDF peers once entering their request window.



(a) Playback continuity. (b) Buffer probability.

Fig. 3: Streaming performance and buffer characteristics (default configuration), comparing EDF, LDF, and SCHEDMIX.

Figure 4 shows the resulting request rates for the individual chunk selection strategies. For the overall population, the request rate drastically drops using the mixed strategy, indicating a major reduction in overhead by roughly 50% for most peers, lending credence to the boon of heterogeneity. When separating strong peers (i.e. peers that play LDF in case of SCHEDMIX) from the rest of the population, it becomes apparent that this reduction is limited to the non-strong sub-population. The strong peers, however, are penalised as their average request rate is slightly increased for SCHEDMIX. At the same time (figures not shown here), the average playback continuity rate across the sub-populations does not show any difference. This supports the argument that there is a high incentive for strong peers to play LDF instead of EDF to improve the overall system performance (see [10] for a game theoretic perspective). These results strengthen our previous analytic arguments and establish that peer heterogeneity can be leveraged to form a powerful mixed scheduling strategy.



(a) All peers. (b) Strong (left) and rest (right).

Fig. 4: Number of requests (default configuration) for (4a) all peers as well separated into strong peers (LDF candidates in SCHEDMIX) and the remaining sub-population (4b).

VI. RELATED WORK

P2P live streaming has been studied extensively in the recent past, albeit in a homogeneous setup. Buffer probability received considerable attention for due reasons. Zhou *et al.* [33], [34] propose a simple model for its analysis based on mean field heuristics. Adamu *et al.* [1] also attempt to analyse it in the context of a discrete Markov chain. Zhao *et al.* [31] develop a population model and make interesting observations about optimal strategies. Outside live streaming, Hajek *et al.* [8], [35] highlight interesting aspects of stability of a P2P system and lay down insightful results on a CTMC formulation. However, there has been little investigation into heterogeneous strategies. The influence of degree, to the best of our knowledge, has also not been studied so far. Our endeavour in this article has been to carefully capture these two important aspects in a principled way. Infection models (see [4], [15], [18]) have proven useful in many computer science problems such as the study of security investments in networks [13], algorithms for distributed systems [5] and particularly many gossip algorithms that later found application in the peer-to-peer area as well [16], [30].

Zhang *et al.* [28] show that pull-based streaming can achieve high bandwidth utilisation and estimate a lower bound for the delivery ratio, based on simulations and a steady-state analysis of simple sender-requester topologies. Liang *et al.* [14] discuss scheduling as a key mechanism for P2P streaming and name source scheduling and bandwidth, the buffer sizes, and degrees as additional factors. Besides, they argue that scheduling plays a role only at a low resource index, whereas we observe clear differences due to content bottlenecks. In [7], a scheduling strategy is proposed, implicitly leveraging heterogeneity by maximising bandwidth utilisation of peers. The authors show a nearly optimal utilisation only for a fully connected mesh, greatly limiting the applicability to realistic setup. In contrast, we focus on pure pull strategies, do not assume a fully connected mesh, and do not focus on maximising bandwidth only. In [19], a mesh/push-based streaming system is proposed using LRU as scheduling strategy. The authors consider overlay rewiring and source scheduling to improve performance and provide supportive experimental results.

VII. DISCUSSION

In this paper, we contributed to building a sound mathematical framework for swarming on random graphs. The dependence of performance on degree was made explicit. The idea of a degree-based (strength-based) combination of primitive scheduling strategies led to two interesting revelations, namely, the boon of heterogeneity and the weak peers outperforming the strong ones. Inspired by these observations, we proposed our mixed strategy SCHEDMIX.

We showed that SCHEDMIX could guarantee good playback continuity at a smaller start-up latency and smaller unsuccessful download rate. The question, however, remains why the strong peers should opt to play LDF. We answer this question with a game theoretic argument in [10] where we established that SCHEDMIX is a Nash equilibrium.

The basic idea behind SCHEDMIX is rather simple: exploit the capabilities of the strong peers to help the weak ones. SCHEDMIX achieves this through degree-based assignment of strategies, but the notion goes beyond degrees. The virtues of

SCHEDMIX can also be achieved, perhaps more pronouncedly, by taking into account other important networking factors such as betweenness centrality, well-connectedness to the server. Our initial simulation results with betweenness centrality-based strategy assignment (not shown here) are affirmative.

We observed that it required only a small percentage of strong peers to uplift the weak peers and improve overall playback experience. However, the optimal percentage of strong peers required to do so is an open research problem.

Our mathematical framework can also serve as a foundation in problems other than the one in pursuit, *e.g.*, network security problems such as circulation of updates to anti-virus in the event of cyber attacks or the circulation of virus/malware itself, supply chain problems for products with limited validity, express consignment delivery problems. Its shifting feature makes it particularly interesting as it allows for multiple interpretations, *e.g.*, advertisement of promotional offers with deadlines, gradual recovery or mutation in the context of infection spread. Keeping analytic tractability aside, the prospect of incorporating more sophisticated mechanisms in practical implementation is broad. We expect to see application of SCHEDMIX in combination with more sophisticated mechanisms. One straightforward but important step is the application of SCHEDMIX in a state-of-the-art hybrid streaming system, where both mesh/pull and multi-tree/push-based mechanisms coexist. In this context it would also be interesting to understand the impact of other mechanisms, such as exchange of buffermaps or a streaming of layered media content. The results presented in this paper are encouraging in that SCHEDMIX could be used as an alternative to complex scheduling strategies in the growing number of scenarios where peer heterogeneity is inevitably given, *e.g.*, when bandwidth-constrained mobile users meet well-connected and high-capacity home users. Besides, the results could be used in the planning of *transitions* [6] between strategies when environmental conditions change.

ACKNOWLEDGEMENT

This work has been funded by the German Research Foundation (DFG) as part of project C03 within the Collaborative Research Center (CRC) 1053 – MAKI. The authors would like to thank Ralf Steinmetz for his valuable input and feedback.

REFERENCES

[1] A. Adamu, Y. Gaidamaka, and A. Samuylov, "Discrete Markov Chain Model for Analyzing Probability Measures of P2P Streaming Network," in *NEW2AN*, ser. LNCS. Springer, 2011, vol. 6869.

[2] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, 1999.

[3] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network*, vol. 14, no. 1, 2000.

[4] R. Durrett, *Random Graph Dynamics*. Cambridge University Press.

[5] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié, "Epidemic Information Dissemination in Distributed Systems," *IEEE Computer*, vol. 37, no. 5, 2004.

[6] A. Frömmgen, B. Richerzhagen, J. Rückert, D. Hausheer, R. Steinmetz, and A. Buchmann, "Towards the Description and Execution of Transitions in Networked Systems," in *AIMS*, 2015.

[7] Y. Guo, C. Liang, and Y. Liu, "AQCS: Adaptive Queue-based Chunk Scheduling for P2P Live Streaming," in *IFIP NETWORKING*, 2008.

[8] B. Hajek and J. Zhu, "The Missing Piece Syndrome in Peer-to-Peer Communication," in *IEEE ISIT*, 2010.

[9] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*. van Nostrand, Princeton, NJ, 1960.

[10] W. R. KhudaBukhsh, J. Rückert, J. Wulfheide, D. Hausheer, and H. Koeppl, "A Comprehensive Analysis of Swarming-based Live Streaming to Leverage Client Heterogeneity," Technische Universitaet Darmstadt, Germany, Tech. Rep., 12 2015. [Online]. Available: http://www.bcs.tu-darmstadt.de/media/bcs/Technical_Report_WKB_1.pdf

[11] D. Krishnappa, M. Zink, and R. Sitaraman, "Optimizing the Video Transcoding Workflow in CDNs," in *ACM MM*, 2015.

[12] T. G. Kurtz, *Approximation of Population Processes*. SIAM, 1981.

[13] M. Lelarge and J. Bolot, "A local mean field analysis of security investments in networks," in *Proceedings of the 3rd International Workshop on Economics of Networked Systems*. ACM, 2008.

[14] C. Liang, Y. Guo, and Y. Liu, "Is Random Scheduling Sufficient in P2P Video Streaming?" in *IEEE ICDCS*, 2008.

[15] T. M. Liggett, *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes*. Springer, 1999.

[16] J. Liu, S. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proceedings of the IEEE*, vol. 96, no. 1, Jan 2008.

[17] Organisation for Economic Co-operation and Development, "OECD Broadband Report," Tech. Rep., 2014.

[18] R. Pastor-Satorras and A. Vespignani, "Epidemic Dynamics in Finite Size Scale-free Networks," *Physical Review E*, vol. 65, no. 3, 2002.

[19] F. Picconi and L. Massoulié, "Is There a Future for Mesh-based Live Video Streaming?" in *IEEE P2P*, 2008.

[20] R. Rejaie and N. Magharei, "On Performance Evaluation of Swarm-based Live Peer-to-Peer Streaming Applications," *Springer Multimedia Systems*, vol. 20, no. 4, 2014.

[21] B. Richerzhagen, D. Stingl, J. Rückert, and R. Steinmetz, "Simonstrator: Simulation and Prototyping Platform for Distributed Mobile Applications," in *ICST/ACM SIMUtools*, 2015.

[22] J. Rückert, B. Richerzhagen, E. Lidanski, R. Steinmetz, and D. Hausheer, "TopT: Supporting Flash Crowd Events in Hybrid Overlay-based Live Streaming," in *IFIP NETWORKING*, 2015.

[23] D. Stingl, C. Gross, J. Rückert, L. Nobach, A. Kovacevic, and R. Steinmetz, "PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems," in *IEEE HPCS*, 2011.

[24] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming," *IEEE TPDS*, vol. 21, no. 3, 2010.

[25] D. J. Watts and S. H. Strogatz, "Collective Dynamics of 'Small-world' Networks," *Nature*, vol. 393, 1998.

[26] M. Wichtlhuber, B. Richerzhagen, J. Rückert, and D. Hausheer, "TRANSIT: Supporting Transitions in Peer-to-Peer Live Video Streaming," in *IFIP NETWORKING*, 2014.

[27] L. Ying, R. Srikant, and S. Shakkottai, "The Asymptotic Behavior of Minimum Buffer Size Requirements in Large P2P Streaming Networks," in *IEEE Information Theory and Applications Workshop (ITA)*, 2010.

[28] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the Power of Pull-based Streaming Protocol: Can We Do Better?" *IEEE JSAC*, vol. 25, no. 9, 2007.

[29] X. Zhang and H. Hassanein, "A Survey of Peer-to-Peer Live Video Streaming Schemes - An Algorithmic Perspective," *Computer Networks*, vol. 56, no. 15, 2012.

[30] X. Zhang, J. Liu, B. Li, and T. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *IEEE INFOCOM 2005*, vol. 3, March 2005.

[31] B. Zhao, J. Lui, and D. Chiu, "Exploring the Optimal Chunk Selection Policy for Data-driven P2P Streaming Systems," in *IEEE P2P*, 2009.

[32] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec, "Peer-Assisted Content Distribution in Akamai NetSession," in *ACM IMC*, 2013.

[33] Y. Zhou, D. M. Chiu *et al.*, "A Simple Model for Analyzing P2P Streaming Protocols," in *IEEE ICNP*, 2007.

[34] Y. Zhou, D.-M. Chiu, and J. Lui, "A Simple Model for Chunk-scheduling Strategies in P2P Streaming," *IEEE/ACM TON*, vol. 19, no. 1, 2011.

[35] J. Zhu and B. Hajek, "Stability of a Peer-to-Peer Communication System," *IEEE Transactions on Information Theory*, vol. 58, no. 7, 2012.

Mistrustful P2P: Privacy-preserving File Sharing Over Untrustworthy Peer-to-Peer Networks

Pedro Moreira da Silva*, Jaime Dias*, Manuel Ricardo*
*INESC TEC, Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal
Email: {pmms, jdias, mricardo}@inesctec.pt

***Abstract*—Peer-to-Peer networks are extensively used for large-scale file sharing. As more information flows through these networks, people are becoming increasingly concerned about their privacy. Traditional P2P file sharing systems provide performance and scalability at the cost of requiring peers to publicly advertise what they download. Several P2P privacy-enhancing systems have been proposed but they still require peers to advertise, either fully or partially, what they download. Lacking alternatives, users have adopted anonymity systems for P2P file sharing, misunderstanding the privacy guarantees provided by such systems, in particular when relaying traffic of insecure applications such as BitTorrent.**

Our goal is to prevent any malicious peer(s) from ascertaining users' content interests so that plausible deniability always applies. We propose a novel P2P file sharing model, Mistrustful P2P, that (1) supports file sharing over open and untrustworthy P2P networks, (2) requires no trust between users by avoiding the advertisement of what peers download or miss, and (3) still ensures deterministic protection of user's interests against attacks of size up to a configured privacy protection level. We hope that our model can pave the ground for a new generation of privacy-enhancing systems that take advantage of the new possibilities it introduces. We validate Mistrustful P2P through simulation, and demonstrate its feasibility.

I. Introduction

Peer-to-Peer (P2P) networks are extensively used for large-scale file sharing. As more information flows through these networks, people are becoming increasingly concerned about their privacy. The reasons behind the privacy concerns may be various such as (1) avoiding user profiling, tracking and data mining, (2) engaging in legal content sharing that may be embarrassing or deplorable from a political, religious or social point-of-view, or (3) engaging in illegal or incriminating content sharing.

Traditional P2P file sharing systems are designed for performance and scalability. These systems take advantage of the large number of interconnected peers[1], and their idle resources, to more efficiently distribute contents at the cost of requiring peers to publicly advertise what they download. Given that peers form interest-based communities [6], every single connection presents an opportunity for a malicious peer to passively obtain additional information that may enable user's content interests identification.

Several P2P privacy-enhancing systems have been proposed, such as [17], [19], [12], [13], [10], the majority employing either techniques to provide anonymity, such as *onion routing* [11] and *information slicing* [13], or employing techniques to provide plausible deniability, such as *request relaying* – peers relay requests to create uncertainty about communicating endpoints –, and *content interest disguise* – peers download additional contents to hide their real interests. All these solutions share one common issue: they require peers to advertise, either fully or partially, what they download. Lacking alternatives, users have adopted anonymity systems for P2P file sharing [4], misunderstanding the privacy guarantees provided by such systems [5], in particular when relaying traffic of insecure applications [14], i.e., applications that disclose sensitive information.

Our objective is to prevent any malicious peer(s) from ascertaining the interests of any user downloading a content, either through observation or through active probing attacks, while completing the download in a timely manner. Users interested in downloading contents are provided with plausible deniability against regular peers or groups of colluding peers.

In this paper, we propose a novel P2P file sharing model, which we name Mistrustful P2P, that enables file sharing over open and untrustworthy P2P networks (networks in which peers should be mistrusted) without disclosing user's interests. Our model does not require trust between users by avoiding the advertisement of what peers are downloading or missing. The Mistrustful P2P model ensures deterministic protection of user's interests from regular peers or groups of colluding peers of size up to a privacy protection level configured by the user. It resorts on erasure coding to avoid advertising what is downloaded. The remaining of this paper is structured as follows. Section II details the problem we aim to solve. Section III presents the related work. Section IV provides the required background. Section V depicts the novel P2P file sharing model we propose. Sections VI and VII describe, respectively, the validation of our model through simulation, and the results obtained. Section VIII presents the conclusions.

II. Problem Description

One privacy aspect that is especially sensitive to users is the concealment of their interests. Users look for a privacy-enhancing system that is able to protect their interests from other participants in the system without compromising performance. Also, providing a configurable per content privacy protection level, supporting untrustworthy P2P networks, and

[1]We say *peer* to refer to the network node, and *user* to refer to the person.

ISBN 978-3-901882-84-5, IFIP Networking 2016

having high performance, are all desirable features because user's privacy requirements are idiosyncratic, they may find themselves in need to join untrustworthy P2P networks, and want to download contents as fast as possible. Above all, users tend to prefer security mechanisms that provide strong defense against well defined attacks, even if narrow ones, rather than broad but weak security defense mechanisms.

We consider an attacker that participates in the system, either a regular peer or a group of colluding peers, but not external entities monitoring all traffic of a peer, such as ISPs, or controlling the whole network, such as governments. Protection against link monitoring could be achieved by encrypting communications between peers, but requires key exchange and distribution mechanisms, which are out of the scope of this work.

P2P privacy-enhancing systems typically either completely hide user's activities through anonymity or disguise them by relaying traffic and/or generating cover downloads. Anonymity systems, being Tor [10] the most popular, as a rule, are not designed for P2P file sharing. Nevertheless, lacking alternatives, users have adopted anonymity systems for such end [4], misunderstanding the privacy guarantees they provide [5], and unaware of the privacy impact that relaying traffic of insecure applications, such as BitTorrent, introduces [14]. On the other hand, systems disguising user's activities are designed for P2P file sharing but require users to publicly advertise what they download, either fully or partially, so that peers know to whom blocks (*chunks*, using BitTorrent terminology) can be requested, and also to improve content availability by providing incentives to download rarer blocks. However, advertising what is downloaded (block advertisement) enables download progress tracking, allowing passive attackers to differentiate genuine from cover traffic, therefore disclosing user's interests. Content interest disguise systems that fully download cover contents are an exception to this, but increase greatly the network overhead.

Thus, the problem we aim to solve is how to enable P2P file sharing so that (1) block advertisement and trust links between users are avoided, (2) users are protected against attacks of size for a privacy protection level that is flexible and configurable per content, and (3) contents can be downloaded in due time.

III. Related Work

Several P2P privacy-enhancing systems have been proposed in the literature providing different degrees of privacy to users, the majority of which provides either anonymity or plausible deniability. Tor and Freenet [1] are probably the most prominent anonymity solutions for, respectively, low-latency anonymity and anonymous content distribution networks. Given that anonymity systems tend to introduce more overhead, and do that without improving the overall performance, herein, we depict the state-of-the-art P2P privacy-enhancing systems providing plausible deniability and designed specifically for P2P file sharing.

BitBlender [3] provides plausible deniability by introducing relay peers that simply proxy requests on behalf of other peers. Peers willing to act as relay peers can register at a central node called *blender*, and, once requested, will join a P2P swarm (group of peers sharing a content) in a probabilistic way so that they cannot be distinguished from regular peers. The joining probability of relay peers is defined by the *blender*, when asking registered peers to join a P2P swarm, so that the set of relay peers remains unknown while having the cardinality requested by the *tracker*. As so, BitBlender requires users to trust both the *tracker* and the *blender*.

SwarmScreen [6] provides plausible deniability by obscuring user's interests through cover traffic (content interest disguise). The devised scheme, which consists in "adding a small percentage (between 25% and 50%) of additional random connections that are statistically indistinguishable from natural ones", thwarts guilt-by-association attacks, i.e., attacks in which the user's interests can be inferred with high certainty just by classifying peers based on the behavior of the communities they participate in. SwarmScreen's attack model only considers passive attacks, it is vulnerable to active attacks.

OneSwarm [12] attempts to be an alternative to BitTorrent, and builds upon friend-to-friend networks – networks in which peers only communicate with trusted peers (friends). It provides a high privacy protection level and extensive control over what information is disclosed to other peers. Nevertheless, content availability may be limited as it is difficult to connect any pair of peers using just trusted links. Also, the problem of providing such privacy guarantees in large groups of untrusted peers remains unsolved.

The BitTorrent Anonymity Marketplace [16] follows SwarmScreen's approach to provide plausible deniability. However, in order to protect against both passive and active attacks, all contents are fully downloaded because peers advertise what they download. The authors define *k-anonymity* as the privacy protection level obtained from fully downloading k contents. Thus, as it increases greatly the network overhead, it either prevents downloads from completing in due time or constrains the privacy protection level.

Petrocco et. al [17], following SwarmScreen's approach, proposed a system that aims to protect user's interests without compromising download completion in due time. Their system relies on private swarms, *request relaying*, caching, and partial advertisement of downloaded blocks. As stated by the authors, private swarms are required to ensure a good level of privacy. Yet, to obtain the credentials needed to join a private swarm, peers must trust one or more participants. Also, as only a fraction of the blocks are advertised, it is not clear how a content sharing is bootstrapped with few seeders nor how request relay should operate during periods of content unavailability.

IV. Erasure Codes

Erasure codes are a class of Forward Error Correction (FEC) codes for the Binary Erasure Channel (BEC), a channel in which transmitted data packets are either correct or missing (erasures). Networking layers above the data link layer behave as an erasure channel since packets are either correct, and are delivered, or present errors, and are discarded.

An erasure code generates a set of n symbols from a set of k symbols, $k < n$, at a rate given by k/n, so that any subset of $k(1+\epsilon(k))$ is enough to reconstruct the original information, where $\epsilon(k)$ is the erasure coding overhead. Erasure codes are usually classified according to three orthogonal properties: (1) systematicity, (2) rate fixedness, and (3) coding overhead. An erasure code is systematic if the input symbols are embed into output symbols, and non-systematic otherwise. If n is static and needs to be known before encoding, the erasure code is fixed-rate. If n can be dynamically increased and the amount of symbols that can be generated does not impose any practical limitation, the erasure code is rateless. Finally, an erasure code is said MDS (Maximum Distance Separable) if any k symbols out of n are enough to reconstruct the original information $[\epsilon(k) = 0]$, or non-MDS if additional symbols are required $[\epsilon(k) > 0]$. Non-MDS erasure codes reduce significantly the encoding and decoding time complexity orders by introducing coding overhead.

For P2P file sharing, MDS erasure codes are more suitable as the network is typically the most constrained resource, not the CPU [15]. Non-systematic erasure codes may have the property of only granting access to any part of a content after fully downloading it. Rateless erasure codes enable the setting of n as a function of hard to predict dynamic variables, such as peer arrival rate, to continuously adjust it to the P2P file sharing dynamics.

V. Mistrustful P2P Model

In this section, we describe the Mistrustful P2P model, a novel P2P file sharing model that (1) supports file sharing over open and untrustworthy P2P networks, (2) requires no trust between users by avoiding the advertisement of what peers download or miss, and (3) still ensures deterministic protection of user's interests, through plausible deniability, against attacks of size up to a configured privacy protection level. We consider that the burden of an increased privacy protection level should be on the peer requiring it and not on other peers' resources, thereby peers communicate through direct links, i.e., there is no peer relaying. For this reason, our model relies on cover downloads to protect user's interests, and therefore, targets the development of content interest disguise systems.

The description of each component of the model is conceptual but we provide the instantiation used for validating the Mistrustful P2P model (Section VI) as an example. We hope that our model can pave the ground for a new generation of privacy-enhancing systems that take advantage of the new possibilities it introduces.

A. Overview

Mistrustful P2P avoids block advertisement, and therefore peers no longer know to whom blocks can be requested nor can request a specific block they need. Consider a content divided into k blocks, and that a block request is sent to a randomly selected peer which offers a randomly selected block it owns. Such approach enables to share some blocks between peers but is unfeasible for fully downloading contents because, assuming an uniform distribution of blocks among peers, the probability of obtaining the last block is just $1/k$. Using erasure codes we are able to generate a set of n blocks so that any subset of k' blocks enables to retrieve the content, where $k' = k(1+\epsilon(k))$, and $\epsilon(k)$ is the erasure coding overhead. As so, for the same conditions, the probability of retrieving the last block increases to $1 - \frac{k'-1}{n}$.

Peers only share erasure coded blocks to ensure that access to any part of a content is only granted after fully downloading it, albeit all contents being publicly available. This way, there is no proof that a user had full or partial access to a particular content, including cover ones, by just having downloaded some blocks; thus, they can still participate in its sharing. We assume that this property is provided by the erasure codes, although content encryption can be used to achieve the same goal. Unless otherwise stated, from now on, we say block to refer to erasure coded block.

The Mistrustful P2P model aims at enabling P2P file sharing in large groups of untrusted peers, thereby, no trust links between peers are required. Attending to the idiosyncrasy of user's privacy requirements and to the flexibility required to not constrain the privacy-enhancing systems that can be built on top of our model, the user privacy protection level is configurable per peer and per content. It is defined as a two-dimensional variable composed by c, the size of the largest colluding group considered by the user, and m, the maximum number of blocks that can be shared with any set of c peers, where $c \leq m$ and $m < k$ so that the content cannot be fully downloaded from a single considered colluding group. Thereby, our model provides a deterministic protection of user's interests as long as the effective size of the largest colluding group does not exceed the one configured. When the user privacy requirements for a particular download are not met, the download pauses until they are met again.

Peers, per content, can take one of two roles depending on their privacy requirements and the way they contribute to the file sharing: *seeder* – peer having a content that wants to share, and willing to forgo its privacy –, or *commoner* – peer willing to download content blocks if its privacy requirements can be met. *Seeders* may be the authors or some party interested in publishing the content, and therefore do not require interest's concealment. We consider that there is always at least one *seeder* to ensure content availability, which provides a new erasure coded block for each request it receives. This is a realistic assumption given that a small fraction of publishers are responsible for 67% of the published content and 75% of the downloads in BitTorrent [7]. *Seeders* only refuse to serve block requests if they have no resources available. On the other hand, *commoners* do not create new blocks and only share them if their privacy remains protected. They can either act as an helper (cover downloads) or as, using BitTorrent terminology, a *leecher* (genuine downloads). *Commoners* keep track of what they have shared with other peers both for privacy enforcement reasons and to avoid offering a block twice to the same peer. They may refuse to serve block requests (1) due to resources constraints, (2) if they have

no useful blocks to offer due to privacy constraints, or (3) due to content disguise strategies. The reason behind is never disclosed. If a block is offered, the requesting peer can then either cancel the request (duplicate block) or proceed with its download (useful block).

Let k be number of blocks required to fully download a content, n the number of unique blocks that can be generated, n_a the number of unique blocks available for download, and D_i the set of unique blocks already downloaded by *commoner* i. *Commoner* i attempts to select a random peer that maximizes the probability of having it offering a block that is not in the set D_i. Given that any subset of k' blocks out of n_a enables to fully retrieve the content, increasing n_a maximizes the probability of a peer obtaining a useful block; which can be achieved by increasing the number of *seeders*. However, determining which block should be offered in reply to a request, when should a request be sent, and to which peer is not trivial. Let us consider $e_{i,j}$ to be the number of blocks that can be exchanged between *commoners* i and j, and E_i the number of blocks that *commoner* i can exchange with all other peers (available requests), which is limited by the privacy constraints. If *commoner* i makes too much block requests, more block requests will fail to retrieve useful blocks and it will run out of available block requests ($E_i = 0$); on the other hand, if *commoner* i makes too few block requests, more block requests could have been sent and the available requests to *commoner* j will still be far from zero once *commoner* j leaves ($e_{i,j} \gg 0$).

We devised three mechanisms which main purpose is to attend the issues stated above. The block selection mechanism is used by *commoners* to determine which block should be offered to a requesting peer. The request backoff mechanism aims at delaying block requests to help maximizing the amount of useful blocks that can be obtained from the available block requests, in the shortest time frame possible. The peer selection mechanism aims at determining the peers that should be selected to minimize the download time.

In sum, the Mistrustful P2P model relies on cover downloads to protect user's interests, and has five main components: erasure codes, the privacy enforcement mechanism, the request backoff mechanism, the peer selection mechanism, and the block selection mechanism. It is out of the scope of this work to provide optimal instantiations of each component. We provide only, as an example, the instantiation used for validating our model.

B. Erasure Codes

Although other erasure codes can be used, we refer the reader to [9] for a rateless MDS construction of Reed-Solomon codes that we developed for our model. These erasure codes are defined over the finite field $\mathbb{F}_{p^2}$, where p is a Mersenne prime ($p = 2^q - 1$), and $n \leq 2^{q+1}$. Their performance was evaluated over $\mathbb{F}_{(2^{31}-1)^2}$, so $n \leq 2^{32}$, and does not impose any constraints to the file sharing. Also, they are non-systematic erasure codes that have the property of only granting access to any part of a content after fully downloading it.

C. Privacy Enforcement

The privacy enforcement mechanism ensures deterministic protection of user's content interests, through plausible deniability, against attacks of size up to a configured privacy protection level. Mistrustful P2P guarantees that any peer or colluding group, with size up to c peers, are unable to (1) prove that the user downloaded a particular content or had full or partial access to it, and to (2) distinguish between cover and genuine downloads by tracking its progress. The user can configure, per content, the size of the largest colluding group to consider, c, and the maximum amount of blocks that can be shared with any set of c peers, m, where $c \leq m$ and $m < k$ so that the content cannot be fully downloaded from a single group of size c. The protection provided is guaranteed as long as the effective size of the largest colluding group does not exceed the one considered by the user. Given that finding the maximum intersection between the set of blocks exchanged with any c peers is an NP-hard problem [18], we devised a conservative yet efficient algorithm to evaluate the numbers of blocks that can still be shared with a peer. The algorithm is divided into two main functions, one to update the counter of blocks shared with a peer (Function 1), and the other to determine the number of blocks that can still be exchanged with a peer (Function 2).

Function 1 Update Blocks Shared

```
                ▷ commoners is an array sorted by blocks shared.
        ▷ blksShared is the max no. of blocks shared w/ c peers.
function IncrementBlocksShared(id)
    i ← commoners.getIndex(id)
    if invalidIndex(i) then                                  ▷ New.
        commoners.push(id)
        commoners.last.blks ← 1
        i ← commoners.getIndex(id)
    else                                                     ▷ Known.
        commoners[i].blks ← commoners[i].blks + 1
        j ← i − 1
        while validIndex(j) do
            blksI ← commoners[i].blks
            blksJ ← commoners[j].blks
            if blksI > blksJ then                            ▷ Still unsorted.
                swap(commoners[i], commoners[j])
                i ← j
                j ← j − 1
            else                                             ▷ Sorted.
                break
            end if
        end while
    end if
    if i < c then                                ▷ Changes on top c peers.
        blksShared ← blksShared + 1
    end if
end function
```

Function 1 relies on an array sorted by the number of blocks shared with a peer, and the maximum number of blocks shared with any set of c peers. The function receives a peer id as a parameter, and starts by checking if any blocks have been

exchanged with that peer. If it is the first one, sets the number of blocks shared to 1. Otherwise, the number of shared blocks is incremented, and, if needed, some elements are swapped until the array is again sorted. The maximum number of blocks shared with any set of c peers is incremented if the update was in one of the top c positions of the array. Function 1 has linear time complexity.

Function 2 Blocks to Share Left

```
                ▷ commoners is an array sorted by blocks shared.
        ▷ blksShared is the max no. of blocks shared w/ c peers.
function BLOCKSHARELEFT(id)
    if c > m or m ≥ k then                                  ▷ Invalid.
        return 0
    end if
    top ← min(c, commoners.length)                          ▷ Top peers.
    left ← m − blksShared − (c − top)
    i ← commoners.getIndex(id)
    if invalidIndex(i) then                                 ▷ New.
        if commoners.length > c then
            left ← left + commoners.last.blks − 1
        end if
        return left
    else                                                    ▷ Known.
        if i ≥ c then
            left ← left + commoners.last.blks
            left ← left − commoners[i].blks
        end if
        return left − 1
    end if
end function
```

Function 2 relies on the same variables as Function 1, and also receives the same parameter. It starts by checking if the configured privacy protection level is invalid. If it is valid, $left$ contains the number of blocks that can still be exchanged, ensuring that at least one block is exchanged with each one of the top c peers. This value needs to be updated if there are already at least c peers and the peer referred by id is outside of that set. Function 2 runs in logarithmic time.

D. Block Selection

The block selection mechanism is used by *commoners* to determine which block should be offered to a requesting peer. It plays an important role on how the blocks end up distributed among peers, affecting the probability of peers obtaining useful blocks. This mechanism ensures that no block is offered twice to the same peer, and determines when requests should be refused due to the lack of useful blocks to share.

Although this mechanism should use content sharing information as input, such as the number of requests that end up canceled (both as source and destination), for validating the model we select blocks randomly due to its simplicity. With Mistrustful P2P model there is no need to suddenly terminate, remove downloads, or stop sharing because the privacy protection level does not depend on the time a peer keeps sharing a content, as long as cover and genuine downloads are treated in the same way.

E. Request Backoff

The request backoff mechanism aims at delaying block requests to help maximizing the amount of useful blocks that can be obtained from the available block requests, in the shortest time frame possible. It does so by constraining the set of peers to which block requests can be sent (eligible peers), and by determining for how long no block requests should be performed. Therefore, as the former is a direct result of individual peer behavior and the latter depends on the swarm behavior, we define the backoff time has a two-dimensional variable that has a per peer and a swarm components. The peer backoff component provides the delay to return a peer to the set of eligible peers while the swarm backoff component provides the delay to perform a new block request.

A block request has five possible outcomes: 1) refusal – the request is refused by the contacted peer, 2) cancellation – the request is canceled by the requester (duplicate block), 3) acceptance – the request is accepted and a block is downloaded, 4) interruption – the request is accepted but the download is interrupted, and 5) disposal – no request is sent due to the lack of eligible peers. Refusal and disposal reveal no information, but all the others do. Cancellation and acceptance reveal that both peers already own that block; interruption reveals that the contacted peer owns that block.

To validate our model, we considered that the peer backoff component is a function of the block transfer time, b_{tt}, and is defined as $\min(\alpha \cdot \lambda^{\tau}, \mu) = \min\left(\frac{b_{tt}}{8} \cdot 2^{\tau}, \frac{k \cdot b_{tt}}{4}\right)$, where α is the peer base backoff time, λ is the exponential factor, τ is the number of consecutive failed requests (all but disposal), and μ is the maximum peer backoff time (25% of download time). The swarm backoff component should be a function of the swarm dynamics to find the proper amount of block requests but, for the sake of simplicity, it is defined as $\beta + \gamma \cdot \tau = 100 + 100 \cdot \tau$, where β is the swarm base backoff time, γ is the scale factor, and τ is the number of consecutive failed request attempts (including disposal).

F. Peer Selection

The peer selection mechanism also helps to maximize the amount of useful blocks of a given content that can be obtained from the available block requests, in the shortest time frame possible, by selecting the peers that return useful blocks in less time. It depends both on the privacy enforcement and on the request backoff mechanisms. The former provides, for a given content, the list of peers to which no further block requests can be sent; the latter provides, also for a given content, which peers are ineligible for the moment, and when can the next block request be performed.

For the validation, given that we considered homogeneous peers and no parallel requests, and also for the sake of simplicity, we select peers randomly.

VI. Validation

This section details the simulation setup, the peer arrival traces used as simulation input, and the use cases considered to validate the Mistrustful P2P model. Given that simulations are

only as good as their models, the simulations were carried out using the ns-3 network simulator [2], which provides realistic network stack and its protocols. Still, the simulation of large scale P2P networks using accurate and realistic models is a complex task. Thereby, to be able to simulate P2P file sharing with thousands of peers using accurate network models, we also use CIDRarchy module [8], a module that we developed for ns-3 that performs IP packet forwarding in constant time.

The validation of our model is done by asserting that peers are able to download contents in due time without advertising what they download. To do so, we simulate the content sharing to evaluate the rate of peers that are able to complete their downloads, and the average download time. Given that the content download due time is subjective, we consider that a content is received in due time if the average download time is, at most, one order of magnitude above direct download time. For the sake of clarity, although cover downloads are required to protect user's content interests, we consider a single content download and no cover downloads. Also, peers are provided with a list of all peers currently in the swarm, request one block at a time, and accept one request at a time. We consider the worst case scenario for how long peers share a given content: peers leave immediately after finishing the download. It is out of the scope of this work to provide a performance comparison with state-of-the-art privacy-enhancing systems.

A. Simulation Setup

We consider a star network topology with a central node mimicking an ISP, and with homogeneous leaf nodes connecting to it through asymmetric links: 30 Mbit/s downlink and 3 Mbit/s uplink. As described in Section V, our model was instantiated as follows. We considered Storm erasure codes, and therefore, any subset of k blocks enables to fully download a content, *seeders* generate a new block per request, and peers only have access to the content after fully downloading it. The privacy mechanism ensures that peers do not exchange more than m blocks with any set of c peers. The block selection, and peer selection mechanisms select, respectively, blocks and peers randomly. The request backoff mechanism sets the peer backoff component as a function of block transfer time that grows exponentially with failed requests, while the swarm backoff component is set as a linear function of failed request attempts.

To ensure that the peer arrival models are realistic, we gathered peer arrival traces of several contents and use them as input to the simulation. The traces were collected by querying a tracker for typical BitTorrent contents, and provide the number of new peers that arrived within ten minute intervals since content publication up to 21 days. We consider the peer arrivals to be independent within each interval, and therefore, we use an exponential function to generate the peer inter-arrival times within that period (Poisson process). We classify content's popularity according to their average peer arrival rate: more popular contents are those that have higher average peer arrival rates. From those collected traces, we selected three video traces for comparison that have different degrees of popularity.

TABLE I
OVERALL NUMBER AND RATIO OF DOWNLOADS COMPLETE.

Contents	1 *Seeder*, Col. of 1	1 *Seeder*, Col. of 32	64 *Seeders*, Col. of 1	64 *Seeders*, Col. of 32
VideoMP_100	75296 (99.82%)	75294 (99.82%)	75323 (99.86%)	75321 (99.85%)
VideoMP_800	74509 (98.78%)	74520 (98.79%)	74565 (98.85%)	74569 (98.86%)
VideoP_100	22444 (99.45%)	22439 (99.42%)	22471 (99.57%)	22472 (99.57%)
VideoP_800	21772 (96.46%)	21762 (96.42%)	21808 (96.63%)	21821 (96.69%)
VideoLP_100	3308 (99.91%)	3257 (98.37%)	3308 (99.91%)	3308 (99.91%)
VideoLP_800	3257 (98.37%)	3240 (97.86%)	3271 (98.79%)	3269 (98.73%)

B. Use Cases

For each individual peer arrival trace we consider eight use cases, which are a result of combining three distinct variables, each taking one of two possible values. We consider a privacy protection level against single peer attacks (collusion of 1) or collusion group attacks of, at most, 32 peers (collusion of 32). Contents are always divided into 64 blocks, have a size of either 100 MiB or 800 MiB, and are shared either by 1 *seeder* or 64 *seeders*; *seeders* are always present during the content sharing. Given that, for the traces we collected, the peer arrival peak usually occurs within the first 36 hours, each use case is simulated for 48 hours to encompass, at least, the content bootstrap and the content sharing peak. We consider $m = k - 1$, i.e., no single peer can download all k blocks from peers belonging to a group of c peers.

Our goal is to validate the model for different content popularities, privacy protection levels, content sizes, and number of *seeders*.

VII. RESULTS AND DISCUSSION

In this section, we present the simulation results for the validation of the Mistrustful P2P model. For each use case, we measured the rate of peers that completed the download, and the average download time. All values are for one hour intervals, thus, for the sake of clarity, we use 'overall' to differentiate between the values for the whole simulation and those for one hour intervals. Figure 1 depicts the number of downloads completed over time, and Table I provides the overall number of downloads completed and the overall completion rate. The average download time is illustrated on Figure 2 while the average overall download time, and the ratio to the direct download time are presented on Table II. Content download is limited by the uplink (3 Mbit/s) because block requests are performed one at a time, thus to a single peer at a time. As so, the reference download times for direct download of 100 MiB and 800 MiB contents are, respectively, approximately 5 and 38 minutes.

The results demonstrate that our model is feasible as peers were able to complete their downloads, and do so in due

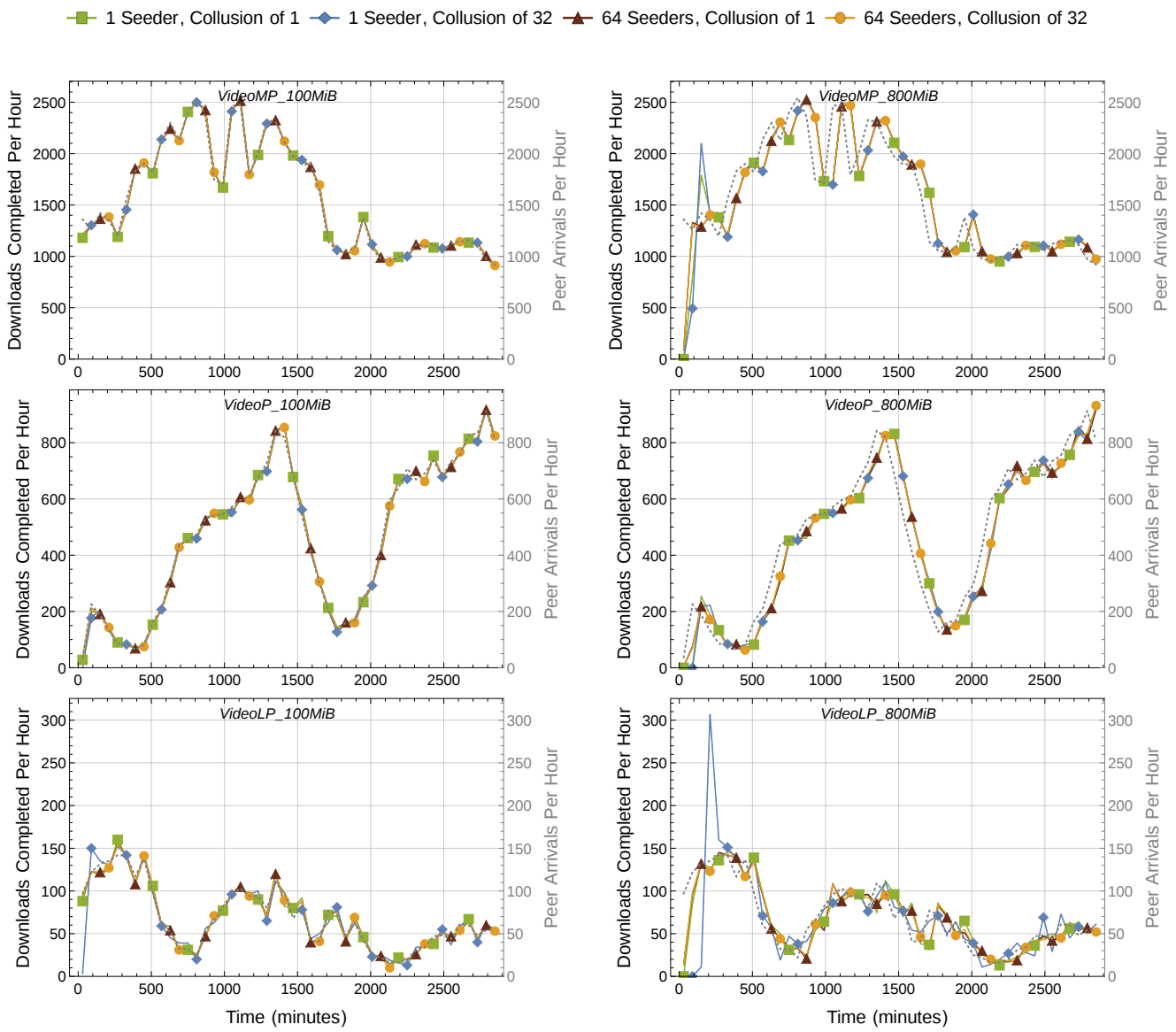


Fig. 1. Number of downloads completed over one hour periods for 100 MiB (left) and 800 MiB (right) contents using a more popular (*MP*), a popular (*P*), and a less popular (*LP*) peer arrival traces as input (one per row). Each plot depicts four use cases that are a result of using either 1 or 64 *seeders*, and considering either single peer attacks or collusion attacks of, at most, 32 peers. The peer arrival rate is represented in gray with a y-scale on the right.

time, without advertising what they have downloaded. For most use cases, the average overall download time is close to the direct download time (see Table II). As shown in Figure 1, peers download completion rate closely follows the peer arrival rate with an offset, which increases as the size of the content increases because peers need to stay longer to fully download the content. Table I shows that the overall download completion rate is very high; the only peers that have not completed the download are those that were sharing when the simulation stopped. Figure 2 shows that peers complete their downloads in due time, and that the average download time depends on the peer arrival rate (content popularity), the privacy protection level, the number of *seeders* sharing the content, and on the content size.

The average download time decreases down to a minimum near the direct download time as the peer arrival rate increases. As the peer arrival rate decreases, both the average download time and the download time variance increase, which suggests that some peers have to wait for others to join before being

TABLE II
AVERAGE OVERALL DOWNLOAD TIME, IN MINUTES, AND RATIO TO DIRECT DOWNLOAD TIME.

Contents	**1 *Seeder*, Col. of 1**	**1 *Seeder*, Col. of 32**	**64 *Seeders*, Col. of 1**	**64 *Seeders*, Col. of 32**
VideoMP_100	8.4 (1.8)	8.4 (1.8)	6.8 (1.5)	6.8 (1.5)
VideoMP_800	60.7 (1.6)	60.8 (1.6)	57.8 (1.6)	57.6 (1.5)
VideoP_100	8.4 (1.8)	11.0 (2.4)	6.0 (1.3)	6.0 (1.3)
VideoP_800	60.1 (1.6)	60.5 (1.6)	55.5 (1.5)	55.4 (1.5)
VideoLP_100	8.4 (1.8)	51.1 (11.0)	5.2 (1.1)	5.2 (1.1)
VideoLP_800	61.0 (1.6)	85.1 (2.3)	47.8 (1.3)	48.0 (1.3)

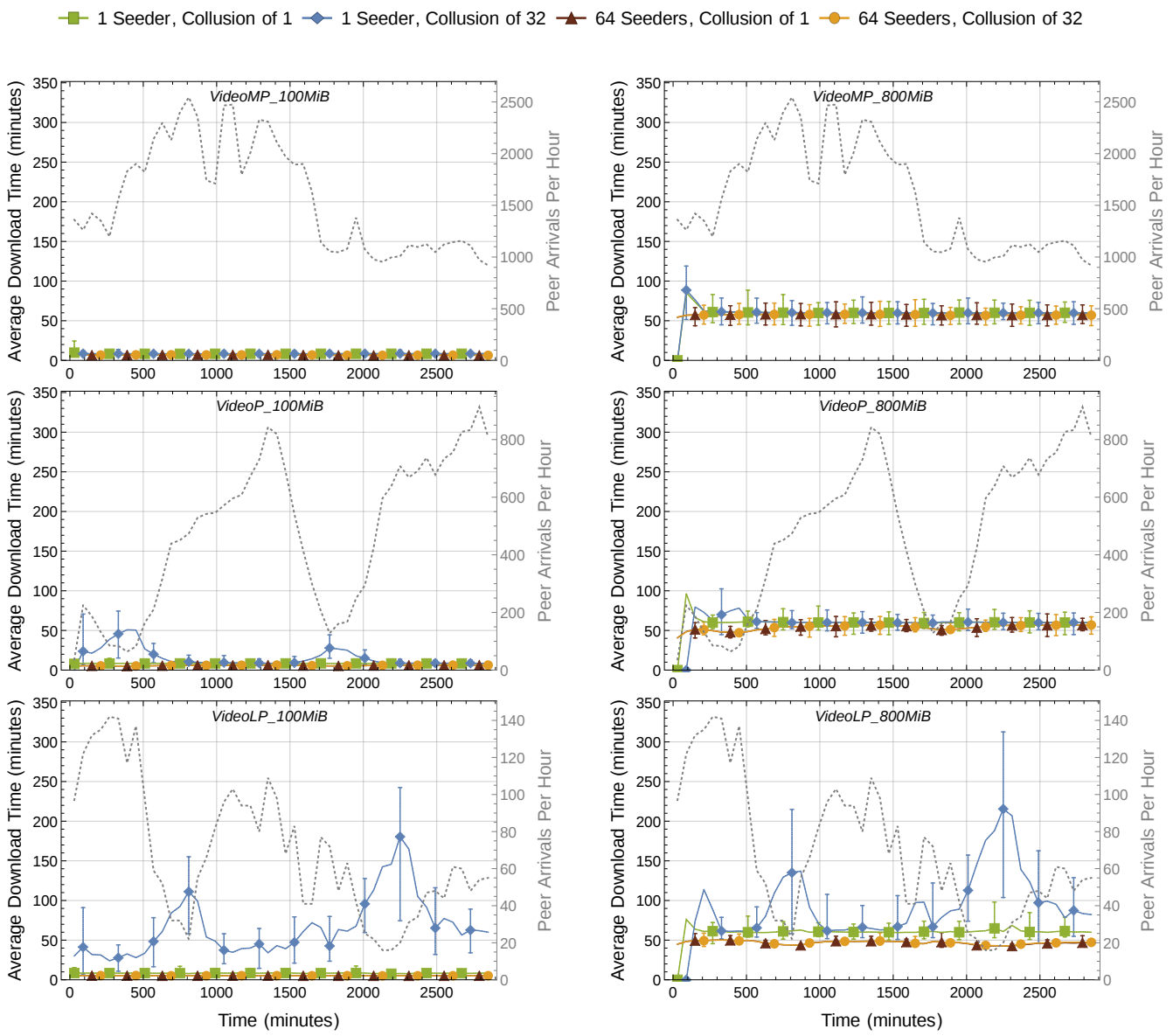


Fig. 2. Average download time over one hour periods for 100 MiB (left) and 800 MiB (right) contents using a more popular (*MP*), a popular (*P*), and a less popular (*LP*) peer arrival traces as input (one per row). Each plot depicts four use cases that are a result of using either 1 or 64 *seeders*, and considering either single peer attacks or collusion attacks of, at most, 32 peers. The bars represent the minimum and maximum download times within one hour intervals. The peer arrival rate is represented in gray with a y-scale on the right.

able to complete the download. The numbers of peers that need to be contacted is constrained by the privacy protection level (at least $c+1$ peers) but also depends on how successful the block requests are, which in turn are dependent on other variables such as the block distribution among the peers. Therefore, the results suggest that the number of peers that need to be contacted is higher than that imposed by the privacy protection level ($c+1$), and the average download time increases when those peers are not immediately available. Adding *seeders* provides a two-fold improvement on the average download time: 1) since *seeders* are always present, less *commoners* need to be simultaneously sharing to be able to complete the download; 2) *seeders* improve the probability of successful block requests as they always offer a useful block, which increases the number of unique blocks available on the network. Unlike direct download time, the average download time does not increase linearly with the increase of the content size. The average number of peers present in the network increases with the increase of the content size because *commoners* have to stay longer to fully download the content, therefore increasing the probability of successful requests, which, in turn, contributes to a lower average download time. This is more evident for less popular contents: for the less popular peer arrival trace, despite the 800 MiB content being eight times larger than the 100 MiB one, the average download time is only less than two times higher.

In sum, the results demonstrate that our model is feasible and, for most of the use cases considered, the average overall download time is close to the direct download time. We considered an instantiation of our model that focus on simplicity instead of optimality, and the peer download completion rate is still very high. For the 64 *seeders* use cases, the average download time is very close to the direct download time, even for a privacy protection level against collusion group attacks of, at most, 32 peers.

VIII. Conclusions

We proposed a novel P2P file sharing model that provides deterministic protection of user's content interests, against attacks of size up to a configured privacy protection level, by avoiding the advertisement of what peers download, as long as the effective size of the largest colluding group does not exceed the one configured; it supports open and untrustworthy P2P networks, and requires no trust links between peers. Our model thwarts passive attacks differentiating genuine from cover downloads using solely block advertisements, and forces attackers to engage in content sharing to know which blocks a peer owns.

By avoiding block advertisement, our model enables peers to control individually what information is disclosed to other peers, and has no requirements on the amount of blocks that have to be downloaded per cover content, so that no single colluding group is able to identify it as a cover content. As so, novel disguise schemes can be devised to conceal user's interests that use more cover contents without increasing the network overhead.

We demonstrated its feasibility through simulation, using ns-3, considering an instantiation of our model focused on simplicity rather than on optimality, and where peers leave immediately after finishing the download. In the majority of the use cases considered, the average overall download time is close to the direct download time. With the Mistrustful P2P model, peers have no need to suddenly terminate or remove downloads because the privacy protection level does not depend on the time a peer keeps sharing a content, as long as cover and genuine downloads are treated in the same way.

As future work, we intend to (1) compare our model against a simple traditional P2P file sharing model, (2) improve the request backoff mechanism to increase the overall performance, and (3) conduct further experiments to evaluate the Mistrustful P2P model with more peer arrival traces, mainly less popular ones, and include more variables such as the number of blocks into which a content is divided. Then, we will analyze how the probability of successful requests changes over time, so that we can improve the instantiation of our model herein presented. We will also propose cover download selection algorithms that minimize the amount of cover traffic required while preserving the privacy protection.

Acknowledgment

This work is financed by the ERDF – European Regional Development Fund – through the Operational Programme for Competitiveness and Internationalisation – COMPETE 2020 Programme – within project "POCI-01-0145-FEDER-006961", and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013 and under the fellowship SFRH/BD/69388/2010.

References

[1] Freenet project. https://freenetproject.org/.
[2] ns-3 network simulator. https://www.nsnam.org/.
[3] K. Bauer, D. McCoy, D. Grunwald, and D. Sicker. BitBlender: Light-weight anonymity for BitTorrent. In *Proceedings of the Workshop on Applications of Private and Anonymous Communications (AlPACa 2008)*. ACM, September 2008.
[4] A. Chaabane, P. Manils, and M. Kaafar. Digging into anonymous traffic: A deep analysis of the Tor anonymizing network. In *Network and System Security (NSS), 4^{th} International Conference on*, pages 167–174, 2010.
[5] S. Chakravarty, G. Portokalidis, M. Polychronakis, and A. Keromytis. Detection and analysis of eavesdropping in anonymous communication networks. *International Journal of Information Security*, 14(3):205–220, 2015.
[6] D. R. Choffnes, J. Duch, D. Malmgren, R. Guiermà, F. E. Bustamante, and L. Amaral. SwarmScreen: Privacy through plausible deniability in P2P systems. Technical report, Northwestern EECS, March 2009.
[7] R. Cuevas, M. Kryczka, A. Cuevas, S. Kaune, C. Guerrero, and R. Rejaie. Is content publishing in BitTorrent altruistic or profit-driven? In *Proceedings of the 6^{th} International COnference*, Co-NEXT '10, pages 11:1–11:12, New York, NY, USA, 2010. ACM.
[8] P. M. da Silva, J. Dias, and M. Ricardo. CIDRarchy: CIDR-based ns-3 routing protocol for large scale network simulation. In *Proceedings of the 8^{th} International Conference on Simulation Tools and Techniques*, SIMUTools '15, pages 267–272, 2015.
[9] P. M. da Silva, J. Dias, and M. Ricardo. Storm: Rateless MDS erasure codes. In *Wireless Internet*, pages 153–158. Springer, 2015.
[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13^{th} Conference on USENIX Security Symposium - Volume 13*, SSYM'04, 2004.
[11] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, Feb. 1999.
[12] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving P2P data sharing with OneSwarm. *SIGCOMM Comput. Commun. Rev.*, 41(4):–, Aug. 2010.
[13] S. Katti, J. Cohen, and D. Katabi. Information slicing: Anonymity using unreliable overlays. In *Proceedings of the 4^{th} USENIX Conference on Networked Systems Design and Implementation*, NSDI'07, 2007.
[14] S. Le Blond, P. Manils, A. Chaabane, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous. One bad apple spoils the bunch: Exploiting P2P applications to trace and profile Tor users. In *Proceedings of the 4^{th} USENIX Conference on Large-scale Exploits and Emergent Threats*, LEET'11, 2011.
[15] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, PODC '02, pages 233–242. ACM, 2002.
[16] S. J. Nielson and D. S. Wallach. The BitTorrent anonymity marketplace. *CoRR*, abs/1108.2718, 2011.
[17] R. Petrocco, M. Capotă, J. Pouwelse, and D. H. Epema. Hiding user content interest while preserving P2P performance. In *Proceedings of the 29^{th} Annual ACM Symposium on Applied Computing*, pages 501–508. ACM, 2014.
[18] M.-Z. Shieh, S.-C. Tsai, and M.-C. Yang. On the inapproximability of maximum intersection problems. *Information Processing Letters*, 112(19):723 – 727, 2012.
[19] P. Tsang, A. Kapadia, C. Cornelius, and S. Smith. Nymble: Blocking misbehaving users in anonymizing networks. *Dependable and Secure Computing, IEEE Transactions on*, 8(2):256–269, March 2011.

Go-with-the-Winner: Performance Based Client-Side Server Selection

Chang Liu[†], Ramesh K. Sitaraman[†‡], and Don Towsley[†]
[†]University of Massachusetts, Amherst [‡]Akamai Technologies Inc.
{cliu, ramesh, towsley}@cs.umass.edu

***Abstract*—Content delivery networks deliver much of the world's web and video content by deploying a large distributed network of servers. We model and analyze a simple paradigm for client-side server selection that is commonly used in practice where each user independently measures the performance of a set of candidate servers and selects the one that performs the best. For web (resp. video) delivery, we propose and analyze a simple algorithm where each user randomly chooses two or more candidate servers and selects the server that provides the best hitrate (resp. bitrate). We prove that the algorithm converges quickly to an optimal state where all users receive the best hitrate (resp. bitrate), with high probability. We also show that if each user chooses just one random server instead of two, some users receive a hitrate (resp. bitrate) that tends to zero. We simulate our algorithm and evaluate its performance with varying choices of parameters, system load, and content popularity.**

I. Introduction

Modern content delivery networks (CDNs) host and deliver a large fraction of the world's web content, video content, and application services on behalf of enterprises that include most major web portals, media outlets, social networks, application providers, and news channels [17]. CDNs deploy large numbers of servers around the world that can store content and deliver that content to users who request it. When a user requests a content item, say a web page or a video, the user is directed to one of the CDN's servers that can serve the desired content to the user. The goal of a CDN is to maximize the performance perceived by the user while efficiently managing its server resources.

A key function of a CDN is *server selection* by which client software running on the user's computer or device, such as media player or a browser, is directed to a suitable server of a CDN [6]. The desired outcome of server selection is that each user is directed to a server that will provide the requested content with good performance. The performance metrics that are optimized vary by the content type. For instance, good performance for a user accessing a web page might mean low latency web page downloads. Good performance for a user watching a video might mean high bitrate video delivery by the server while avoiding video freezing and rebuffering [11].

Server selection can be performed in two distinct ways that are not mutually exclusive. *Network-side server selection* algorithms monitor the real-time characteristics of the CDN and the Internet. Such algorithms are often complex and measure liveness and CDN server load, as well as latency, loss, and bandwidth of the communication paths between servers and users. Using this information, the algorithm computes a good "mapping" of users to servers, such that each user is assigned a "proximal" server capable of serving that user's content [17]. This mapping is computed periodically and is typically made available to the client using the domain name system (DNS). Specifically, the user's browser or media player looks up the domain name of the content that it wants to download and receives as translation the IP address of the selected server.

A complementary approach to network-side server selection that is commonly is used is *client-side server selection* where the client embodies a server selection algorithm. The client software is typically unaware of the global state of the server infrastructure, the Internet, or other clients. Rather, the client software typically makes future server selection decisions based on its own historical performance measurements from past server downloads. Client-side server selection can often be implemented as a plug-in within media players, web browsers, and web download managers [2].

While client-side server selection can be used to select servers within a single CDN, it can also be used in a multi-CDN setting. Large content providers often make the same content available to the user via multiple CDNs. In this case, the client tries out the different CDNs and chooses the "best" server from across multiple CDNs. For instance, NetFlix uses three different CDNs and the media player incorporates a client-side server selection algorithm to choose the "best" server (and the corresponding CDN) using performance metrics such as achievable video bitrates [1]. Note also that in the typical multi-CDN case, both network-side and client-side server selection can be used together, where the former is used to choose the candidate servers from each CDN and the latter is used by the user to pick the "best" among all the candidates.

A. The Go-With-The-Winner paradigm

A common and intuitive paradigm that is often used for client-side server selection in practice is what we call *"Go-With-The-Winner"* that consists of an initial *trial period* during which each user independently "tries out" a set of *candidate servers* by requesting content or services from them (cf. Figure 1). Subsequently, each user independently *decides* on the "best" performing server using historical performance information that the user collected for the candidate servers during the trial period. It is commonly implemented in the

ISBN 978-3-901882-83-8

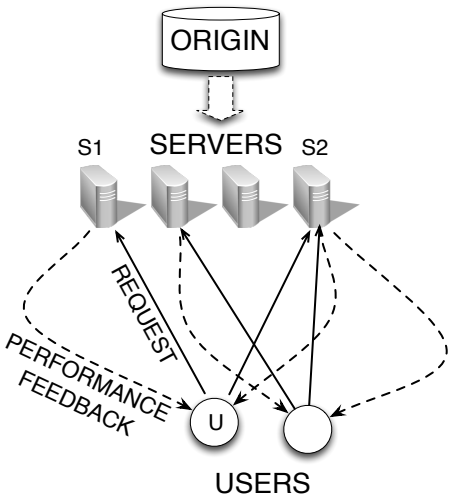


Fig. 1: Client-side Server Selection with the Go-With-The-Winner paradigm. User U makes request to two candidate servers $S1$ and $S2$. After a trial period of observing the performance provided by the candidate, the user selects the better performing server.

content delivery context that incorporate selecting a web or video content server from among a cluster of such servers.

Besides content delivery, the Go-With-The-Winner paradigm is also used for other Internet services, though we do not explicitly study such services in our work. For instance, BIND, which is the most widely deployed DNS resolver (i.e., DNS client) on the Internet, tracks performance as a smoothed value of historical round trip times (called SRTT) from past queries for a set of candidate name servers. BIND then chooses a particular name server to query in part based on the computed SRTT values [12]. It is also notable that BIND implementations incorporate randomness in the candidate selection process.

The three key characteristics of the Go-With-The-Winner paradigm are as follows.

1) *Distributed control.* Each user makes decisions in a distributed fashion using only knowledge available to it. There is no explicit information about the global state of the servers or other users, beyond what the user can infer from it's own historical experience.
2) *Performance feedback only.* There is no explicit feedback from a server to a user who requested service beyond what can be inferred by the performance experienced by the user.
3) *Choosing the "best" performer.* The selection criteria is based on historical performance measured by the user and consists of selecting the best server according to some performance metric (i.e., go with the winner).

Besides its inherent simplicity and naturalness, the paradigm is sometimes the only feasible and robust solution. For instance, in many settings, the client has no detailed knowledge of the state of the server infrastructure as it is managed and owned by other business entities. In this case, the primary feedback mechanism for the client is its own historical performance measurements.

While client-side server selection is widely implemented, its theoretical foundations are not well understood. A goal of our work is to provide such a foundation in the context of web and video content delivery. *It is not our intention to model a real-life client-side server selection process in its entirety which can involve other adhoc implemention-specific considerations. But rather we abstract an analytical model that we can explore to extract basic principles of the paradigm that are applicable in a broad context.*

B. Our contributions

We propose a simple theoretical model for the study of client-side server selection algorithms that use the Go-With-The-Winner paradigm. Using our model, we answer foundational questions such as how does randomness help in the trial period when selecting candidate servers? How many candidate servers should be selected in the trial phase? How long does it take for users to narrow down their choice and decide on a single server? Under what conditions does the selection algorithm converge to a state where all users have made correct server choices, i.e., selected servers provide good performance to their users? Some of our key results that help answer these questions follow.

(1) In Section II, in the context of web content delivery, we analyze a simple algorithm called GoWithTheWinner where each user independently selects two or more random servers as candidates and decides on the server that provides the best cache hit rate. We show that with high probability, the algorithm converges quickly to a state where no cache is overloaded and all users obtain a 100% hit rate. Furthermore, we show that two or more random choices of candidate servers are necessary, as just one random choice will result in some users (and some servers) incurring cache hit rates that tend to zero, as the number of users and servers tend to infinity. This work represents the first demonstration of the "power of two choices" phenomena in the context of client-side server selection for content delivery, akin to similar phenomena observed in balls-into-bins games [14], load balancing, circuit-switching algorithms [4], relay allocation for services like Skype [16], and multi-path communication [10].

(2) In Section III, in the context of video content delivery, we propose a simple algorithm called MaxBitRate where each user independently selects two or more random servers as candidates and decides on the server that provides the best bitrate for the video stream, We show that with high probability, the algorithm converges quickly to a state where no server is overloaded and all users obtain the required bitrate for their video to play without freezes. Further, we show that two or more random choices of candidate servers are necessary, as just one random choice will result in some users receiving bitrates that tend to zero, as the number of users and servers tends to infinity.

(3) In Section IV, we go beyond our theoretical model and simulate algorithm GoWithTheWinner in more complex settings. We establish an inverse relationship between the length of the history used for hitrate computation (denoted by τ) and the failure rate defined as the probability that the system converges to a non-optimal state. We show that as τ increases the convergence time increases, but the failure rate decreases. We also empirically evaluate the impact of the number of choices of candidate servers. We show that two or more random choices are required for all users to receive a 100% hitrate. Though even if only 70% of the users make two choices, it is sufficient for 95% of the users to receive a 100% hitrate. Finally, we show that the convergence time increases with system load. But, convergence time decreases when the exponent of power law distribution that describes content popularity increases.

II. Hit Rate Maximization for Web Content

The key measure of web performance is *download time* which is the time taken for a user to download a web object, such as an html page or an embedded image. CDNs enhance web performance by deploying a large number of servers in access networks "close" to the users. Each server has a cache capable of storing web objects. When a user requests an object, such as a web page, the user is directed to a server that can serve the object (cf. Figure 1). If the server already has the object in its cache, i.e, the user's request is a *cache hit*, the object is served from the cache to the user. In this case, the user experiences good performance, since the CDN's servers are proximal to the user and the object is downloaded quickly. However, if the requested object is not in the server's cache, i.e., the user's request is a *cache miss*, then the server first fetches it from the origin, places it in its cache, and then serves the object to the user. In the case of a cache miss, the performance experienced by the user is often poor since the origin server is typically far away from the server and the user. In fact, if there is a cache miss, the user would have been better off not using the CDN at all, since downloading the content directly from the content provider's origin would likely have been faster! Since the size of a server's cache is bounded, cache misses are inevitable. A key goal of server selection for web content delivery is to jointly orchestrate server assignment and content placement in caches such that the cache hit rate is maximized. While server selection in CDNs is a complex process [17], we analytically model the key elements that relate to content placement and cache hit rates, leaving other factors that impact performance such as server-to-user latency for future work.

A. Problem Formulation

Let U be a set of n_u users who each requests an object picked independently from a set C of size n_c using a popularity distribution $\{p_1, p_2, \ldots, p_{n_c}\}$, where the k-th most popular object in C is picked with probability p_k. The user then makes a sequence of requests for that content item to the set of available servers. In practice, users tend to stay with one website for a while, say reading the news or looking at a friend's posts. We model the sequence of requests generated by each user as a Poisson process with homogeneous arrival rate λ. Note that each request from user u can be sent to one or more servers selected from $S_u \subseteq S$, where S_u is the set of candidate servers for user u.

Let S be the set of n_s servers that are capable of serving content to the users. Each server can cache at most κ objects and a cache replacement policy such as LRU is used to evict objects when the cache is full. Given that the download time of a web object is significantly different when the request is a cache hit versus a cache miss, we make the assumption that the user can reliably infer if its request to download an object from a server resulted in a cache hit or a cache miss immediately after the download completes.

The objective of client-side server selection is for each user $u \in U$ to independently select a server $s \in S$ using only the performance feedback obtained on whether each request was a hit or a miss. Let the hit rate function $H(u, s, t)$ denote the probability of user u receiving a hit from server $s \in S_u$ at time t. We define the system-wide performance measure $H(t)$, as the best hit rate obtained by the worst user at time t,

$$H(t) \triangleq \min_{u \in U} \max_{s \in S_u} H(u, s, t), \tag{1}$$

a.k.a. the *minmax hit rate*. Our goal is to maximize $H(t)$. In the rest of the section, we describe a simple "Go-With-The-Winner" algorithm for server selection and show that it converges quickly to an optimal state, with high probability.

Note: Our formulation is intentionally simple so that it can model a variety of other situations in web content delivery. For instance, a single server could in fact model a cluster of front-end servers that share a single backend object cache. A single object can model a bucket of objects that cached together as is often done in a CDN context [17].

B. The GoWithTheWinner Algorithm

After each user $u \in U$ selects a content item and a set of σ servers S_u, the user executes algorithm *GoWithTheWinner* to select a server likely to always have the content. In this algorithm, each user locally executes a simple "Go-With-The-Winner" strategy of trying out σ randomly chosen candidate servers initially. For each server $s \in S_u$, the user keeps track of the most recent request results in a vector $\mathbf{h}^s = (h_1^s, h_2^s, \cdots, h_\tau^s)$ where $h_k^s = 1$ corresponds to the k-th recent request resulting in a hit from server s and $h_k^s = 0$ if otherwise. τ is the "sliding window size". Using the hit rates, each user then independently either chooses to continue with all the servers in S_u or decides on a single server that provided good performance. If there are multiple servers providing 100% hit rate, the user decides to use the first one found.

C. Analysis of Algorithm GoWithTheWinner

Here we analyze the case where $n_u = n_c = n_s = n$ and experimentally explore other variants where n_c and n_u are larger than n_s in Section II-D and IV. Let $H(t)$ be as defined in (1). If $\sigma \geq 2$, we show that with high probability $H(t) = 100\%$,

Algorithm 1: GoWithTheWinner

```
The current user u chooses a set of σ candidate servers S_u ⊆ S uniformly at random from all the servers;
for each s ∈ S_u do
    set h^s ← (h_1^s, h_2^s, ··· , h_τ^s) = 0;
end
for each arrival of request do
    set t to the current time;
    Request content a_u from all servers s ∈ S_u;
    for each server s ∈ S_u do
        h_i^s ← h_{i-1}^s, 2 ≤ i ≤ τ;
        h_1^s ← if hit; h_1^s ← 0, if miss;
        compute hit rate H_τ(u, s, t) ← (Σ_{i=1}^{τ} h_i^s)/τ ;
        if H_τ(u, s, t) = 100% then
            decide on server s by setting S_u ← {s};
            return;
        end
    end
end
```

for all $t \geq T$, where $T = O(\frac{\kappa}{\log(\kappa+1)}(\log n)^{\kappa+1}\log\log n)$. That is, the algorithm converges quickly with high probability to an optimal state where *every* user has decided on a single server that provides a 100% hit rate, and *every* server has the content requested by its users.

Definitions. A server s is said to be *overbooked* at some time t if users request more than κ distinct content items from server s, where κ is the number of content items a server can hold. Note that a server may have more than κ users and not be overbooked, provided the users collectively request a set of κ or fewer content items. Also, note that a server that is overbooked at time t is overbooked at every $t' \leq t$ since the number of users requesting a server can only remain the same or decrease with time. Finally, a user u is said to be *undecided* at time t if $|S_u| > 1$ and is said to be *decided* if it has settled on a single server to serve its content and $|S_u| = 1$. Note that each user starts out undecided at time zero, then decides on a server at some time t and remains decided in all future time later than t. Users calculate the hit rates of each of the available servers based on a history record of the last τ requests, where τ is called the sliding window size.

Lemma 1: If the sliding window size $\tau = \Theta(\log^{\kappa+1} n)$, the probability that some user $u \in U$ decides on an overbooked server $s \in S_u$ upon any request arrival is at most $1/n^{\Omega(1)}$.

Proof: If user u decides on server s then the current request together with the previous $\tau - 1$ requests are all hits. Let H_k, $k = 1, 2, \cdots, \tau$ be Bernoulli random variables, s.t. $H_k = 1$ if the most recent k-th request of u is a hit and $H_k = 0$ if it is a miss. To prove Lemma 1 we need to show

$$\mathbb{P}\left(\cap_{k=1}^{\tau}(H_k = 1)\right) \leq n^{-\Omega(1)}. \tag{2}$$

Let t_1 denote the time of the most recent request for content a_u from user u appears at server s, resulting in feedback H_1 to the user. Let $t_1 - \Delta$ be the time that the previous request for a_u arrives at s. Let $A_s = \{a_1, a_2, \cdots, a_M\}$ be the set of different content items requested at s, where $M > \kappa$. Let $N_i \geq 1$ be the number of users requesting a_i from s. WLOG, let $a_1 = a_u$ be the content that u requests, such that N_1 is the number of users requesting for a_u. Because we assume all the users generates requests with a Poisson process with arrival rate λ, the aggregated arrival rate of requests for a_u is then $N_1\lambda$. Thus Δ is an exponential random variable, $\Delta \sim Exp(N_1\lambda)$. Now we look at the number of different requests arrives between time $t_1 - \Delta$ and t_1. Let $X_i, i = 2, 3, \cdots, M$ be an indicator that a request for a_i arrives at server s during the time interval $(t_1 - \Delta, t_1)$, we have $X_i \sim Bernoulli(1 - e^{-N_i\lambda\Delta})$. Furthermore, let random variable $Y = \sum_{i=2}^{M} X_i$ be the number of different requests arrived in the time interval. With the server running on LRU replacement policy,

$$\mathbb{P}\left(H_1 = 0\right) = \mathbb{P}\left(Y \geq \kappa\right), \tag{3}$$

because for content a_u to be swapped out of the server, more than κ different requests other than that for a_u must have arrived. Equation (3) shows that H_1 only depends on the number different requests arrived after the previous request for a_u, which means events $H_k, k = 1, 2, \cdots, \tau$ are mutually independent.

Furthermore[1], because $N_i \geq 1$, we have $X_i \geq_d X'$ where $X' \sim Bernoulli(1 - e^{-\lambda\Delta})$. Thus,

$$Y = \sum_{i=2}^{M} X_i \geq_d \sum_{i=2}^{M} X' = Z,$$

where $Z \sim Binomial(\kappa, (1 - e^{-\lambda\Delta}))$.

Thus, we have

$$\begin{aligned}
\mathbb{P}\left(Y \geq \kappa\right) &\geq \mathbb{P}\left(Z \geq \kappa\right) \\
&= \int_0^{\infty} \mathbb{P}\left(Z \geq \kappa | \Delta = t\right) f_{\Delta}(t) dt \\
&= \int_0^{\infty} (1 - e^{-\lambda t})^{\kappa} N\lambda e^{-N\lambda t} dt \\
&= \frac{N!\kappa!}{(N+\kappa)!} \\
&\geq (N+\kappa)^{-\kappa},
\end{aligned}$$

where $f_{\Delta}(t)$ is the probability density function of Δ.

Note that N is the number of users requesting a at server s, and is bounded by $N = O(\frac{\log n}{\log\log n})$, with high probability [19].

Now, we can finally prove (2). Let c' be an appropriate constant,

$$\begin{aligned}
\mathbb{P}\left(\cap_{k=1}^{\tau}(H_k = 1)\right) &= \mathbb{P}\left(H = 1\right)^{\tau} = (1 - \mathbb{P}\left(H = 0\right))^{\tau} \\
&= (1 - \mathbb{P}\left(Y \geq \kappa\right))^{\tau} \\
&\leq (1 - (N+\kappa)^{-\kappa})^{\tau} \\
&\leq (1 - (c'\frac{\log n}{\log\log n} + \kappa)^{-\kappa})^{\tau},
\end{aligned}$$

[1] random variables $U \geq_d V$ if $\mathbb{P}\left(U > x\right) \geq \mathbb{P}\left(V > x\right)$ for all x.

which is $n^{-\Omega(1)}$ when $\tau = \Theta(\log^{\kappa+1} n)$. ■

By bounding the time for τ requests to arrive at user u, we have the following,

Lemma 2: If user u (with candidate servers S_u) is not *decided* at time t, then the server is *overbooked* at time $t - \delta$ for $\delta = \frac{\tau+1}{\lambda} c_0$ where $c_0 > 1$ is a constant, with high probability.

Proof: Let random variable N_δ be the number of requests from u during time $(t-\delta, t)$, $N_\delta \sim Poisson(\lambda\delta)$. A bound on the tail probability of Poisson random variables is developed in [15] as

$$\mathbb{P}(X \le x) \le \frac{e^{-\lambda'}(e\lambda')^x}{x^x},$$

where $X \sim Poisson(\lambda')$ and $x < \lambda'$.

We can show there are at least $\tau+1$ requests during $(t-\delta, t)$ w.h.p. as the following,

$$\begin{aligned}\mathbb{P}(N_\delta < \tau+1) &\le e^{-\lambda\delta}\frac{(e\lambda\delta)^{\tau+1}}{(\tau+1)^{\tau+1}} = e^{-(\tau+1)c_0}(ec_0)^{(\tau+1)} \\ &= e^{-(\tau+1)(c_0-1)}c_0^{(\tau+1)} \\ &= n^{-\frac{(\tau+1)}{\log n}(c_0-1-\log c_0)} \\ &= n^{-\Theta(\log^\kappa n)},\end{aligned}$$

as $c_0 > 1$ and $\tau = \Theta(\log^{\kappa+1} n)$. Thus, w.h.p. no fewer than $\tau+1$ requests arrive at u. And because user is not decided at time t we know that with high probability, at least one of the previous τ requests results in a miss, which means that between the previous $(\tau+1)$-th request and the miss, κ different other requests arrived at the server. Thus server s is *overbooked* at the time the previous $(\tau+1)$-th request arrives, which with high probability is no earlier than $t - \delta$. ■

Based on Lemmas 1 and 2, we can then establish the following theorem about the performance of Algorithm *GoWithTheWinner*.

Theorem 3: With probability at least $1 - \frac{1}{n^{\Omega(1)}}$, the minmax hit rate $H(t) = 100\%$ for all $t \ge T$, provided $\sigma \ge 2$ and $T = O(\frac{\kappa}{\log(\kappa+1)}(\log n)^{\kappa+1}\log\log n)$. That is, with high probability, algorithm *GoWithTheWinner* converges by time T to an optimal state where each user $u \in U$ has decided on a server $s \in S$ that serves it content with a 100% hit rate.

This is the main result for the performance analysis of the algorithm. Due to space limit, please refer to our technical report [13] for detailed proof of this theorem.

Are two or more random choices necessary for all users to receive a 100% hit rate? Analogous to the "power of two choices" in the balls-into-bins context [14], we show that two or more choices are required for good performance with the following theorem.

Theorem 4: For any fixed constants $0 \le \alpha < 1$ and $\kappa \ge 1$, when algorithm GoWithTheWinner uses one random choice for each user ($\sigma = 1$), the minmax hit rate $H(t) = o(1)$, with high probability, i.e., $H(t)$ tends to zero as n tends to infinity, with high probability.

The reader is referred to technical report [13] for the proof.

D. When $n_u = n_s^\alpha, \alpha > 1$

Now we analyze the case that there are many more users than the number of servers. Assume $n_s = n, n_u = n^\alpha$ and $\kappa = \frac{n_u}{n_s} = n^{\alpha-1}$, we have the following result,

Theorem 5: When $n_s = n, n_u = n^\alpha, \alpha > 1$, with probability at least $1 - \frac{1}{n^{\Omega(1)}}$, the maximum load (number of incoming servers) over all servers is $O(\sigma\frac{n_u}{n_s})$. Furthermore, if $\kappa = \frac{n_u}{n_s}$, all users have 100% hit rate.

Theorem 5 implies that when $n_u = n_s^\alpha$ all the servers have balanced load of $\sigma\frac{n_u}{n_s}$, and thus we don't need a server selection mechanism for load balancing other than just letting users randomly choose the server. In this case, it's not beneficial to let users start with more than one randomly selected servers, because with $\sigma = 1$ the load on all servers are balanced already. Thus, as long as we have feasible server capacity $\kappa = \omega(\frac{n_u}{n_s})$, all the users will have enough resources from the server and have 100% hit rate by randomly select one server.

The number of content items n_c here does not affect the result of load balancing. Actually, the result stays the same when $n_c \ge n_u$. When the number of content items is much smaller than number of users, $n_c << n_u$, the cache size can be made smaller because the number of distinct requests at each server becomes smaller.

III. Bitrate Maximization for Video Content

In video streaming, a key performance metric is the *bitrate* at which a user can download a video. If the server is unable to provide the required bitrate to the user, the video may frequently freeze resulting in an inferior viewing experience and reduced user engagement [11]. For simplicity, we model the server's bandwidth capacity that is often the critical bottleneck resource, while leaving other factors that could influence video performance such as the server-to-user connection and the server's cache[2] for future work.

A. Problem formulation

The bitrate required to play a stream without freezes is often the encoded bitrate of the stream. For simplicity, we assume that each user requires a bitrate of 1 unit for playing its video and each server has the capacity to serve an aggregation of κ units. We also assume each server evenly divides its available bitrate capacity among all users streaming videos from it. We assume each user can tell the exact bitrate that it receives from its chosen candidate servers and that this bitrate is used as the performance feedback (cf. Figure 1).

Unlike web content delivery, where users make random requests to the same website, we assume that users requesting video streaming maintain persistent connections with the server. We use a discrete time model in this case as compared to the continuous time model for web content delivery. We assume after each time unit that users examine the bit rate provided by each of the available servers and then make

[2]Unlike the web, cache hit rate is a less critical determinant of video performance. Videos are cached in chunks by the server. The next chunk is often prefetched from origin if it is not in cache, even while the current chunk is being played by the user, so as to hide the origin-to-server latency.

decisions according to the performance (measured by bit rate). The goal of each user is to find a server that can provide the required bitrate of 1 unit for viewing the video.

B. Algorithm MaxBitRate

After each user $u \in U$ has selected a video object $c_u \in C$ using the popularity distribution, Algorithm MaxBitRate described below is executed independently by each user $u \in U$, in discrete time steps.

1) Choose a random subset of candidate servers $S_u \subseteq S$ such that $|S_u| = \sigma$.
2) At each time step $t \geq 0$, do the following:
   a) Request the video content from all servers $s \in S_u$.
   b) For each server $s \in S_u$, compute $B(u,s,t) \triangleq$ bitrate provided by server s to user u in the current time step.
   c) If there exists a server $s \in S_u$ such that $B(u,s,t) = 1$, then *decide* on server s by setting $S_u \leftarrow \{s\}$.

Note that each user executes a simple strategy of trying σ randomly chosen servers initially. Then, using the bitrate received in the current time step as feedback, each user independently narrows it's choice of servers to a single server that provides the required unit bitrate. If multiple servers provide the required bitrate, the user selects one at random. Further, note that a user u downloading from a server s at time t knows immediately whether or not the server is overloaded, since server s is overloaded if user u received a bitrate of less than 1 unit from the server, i.e., $B(u,s,t) < 1$. This is a point of simplification in relation to the complex situation of hit rate maximization where any single cache hit is not indicative of a non-overloaded server and a historical average of hit rates over a large enough time window τ is required as a probabilistic indicator of server overload. Furthermore, this simplification yields both faster convergence to an optimal state in $T = O(\log\log n/\log(\kappa+1))$ steps and a much simpler proof of convergence.

C. Analysis of Algorithm MaxBitRate

As before, we rigorously analyze the case where $n_u = n_s = n$. Let the *minmax* bitrate $B(t)$ be the best bitrate obtained by the worst user at time t, i.e.,

$$B(t) \triangleq \min_{u \in U} \max_{s \in S} B(u,s,t).$$

***Theorem** 6:* When $\sigma \geq 2$, the minmax bitrate converges to $B(t) = 1$ unit, for all $t \geq T$, within time $T = O(\log\log n/\log(\kappa+1))$, with high probability. When $\sigma = 1$ on the other hand, the minmax bitrate $B(t) = O(\kappa \log\log n/\log n)$, with high probability. In particular, when $\sigma = 1$ and the cache size κ is $o(\log n/\log\log n)$, including the case when κ is a fixed constant, $B(t)$ tends to zero as n tends to infinity, with high probability.
The proof can be found in [13].

IV. Empirical Evaluation

We empirically study our algorithm *GoWithTheWinner* through simulation. Requests from each user is modeled as a Poisson arrival sequence with unit rate. We use $n_u = 1000$ users. To simulate varying numbers of servers, users, and content items, we vary n_s and n_c such that $1 \leq n_u/n_c, n_u/n_s \leq 100$. We also simulate a range of values for the spread $1 \leq \sigma \leq 6$, and sliding window size $1 \leq \tau \leq 20$. Each server implements an LRU replacement policy of size $\kappa \geq 2$. We use the power law distribution for content popularity distribution, where the k^{th} most popular object in C is picked with a probability

$$p_k \triangleq \frac{1}{k^\alpha \cdot \mathcal{H}(n_c,\alpha)}, \tag{4}$$

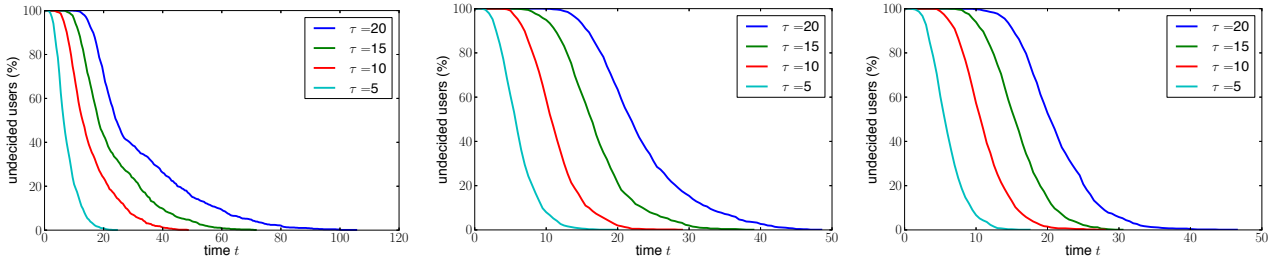
where $\alpha \geq 0$ is the exponent of the distribution and $\mathcal{H}(n_c,\alpha) = \sum_{k=1}^{n_c} 1/k^\alpha$. Note that power law distributions (aka Zipf distributions) are commonly used to model the popularity of online content such as web pages, and videos. This family of distributions is parametrized by a Zipf rank exponent α with $\alpha = 0$ representing the extreme case of an uniform distribution and larger values of α representing a greater skew in the popularity. It has been estimated that the popularity of web content can be modeled by a power law distribution with an α in the range from 0.65 to 0.85 [3], [9], [8]. In the simulations, the content items are requested by users using the power law distribution of (4) with $\alpha = 0.65$ to model realistic content popularity [3] [9]. However, we also vary α from 0 (uniform distribution) to 1.5 in some of our simulations.

The system *converges* when all users have decided on a single server from their set of candidate servers. There are two complementary metrics that relate to convergence. *Failure rate* is the probability that the system converged to a non-optimal state where there exists servers that are overbooked, resulting in some users incurring cache misses after convergence. The failure rate is calculated from multiple runs of the simulation. *Convergence time* is the time it takes for the system to converge provided that it converges to an optimal state.

A. Speed of convergence

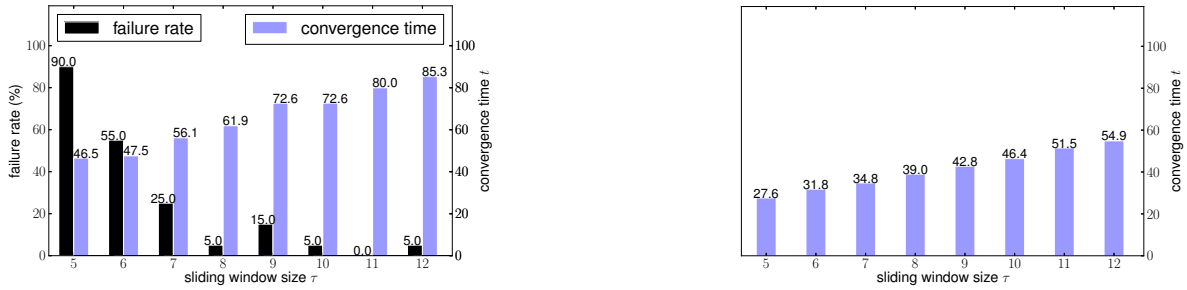
Figure 2 shows how the fraction of undecided users decreases over time until it reaches zero, resulting in convergence. Note that users do not decide in the first τ steps, since they must wait at least that long to accumulate a window of τ hits. However, once the first τ steps complete, the decrease in the number of undecided users is fast as users discover that at least one of their two randomly chosen candidate servers have less load. The rate of decrease in undecided users decreases towards the end, as the number of users who experience cache contention in *both* of their server choices require multiple iterations to resolve.

In this simulation, we keep the number of users $n_u = 1000$ but vary the number of servers n_s to achieve different values for n_u/n_s. Note that for a fair comparison, we keep the system-wide load the same. Load l is a measure of cache



(a) $\alpha = 0.65$, $n_u/n_s = 1$ (b) $\alpha = 0.65$, $n_u/n_s = 10$ (c) $\alpha = 0.65$, $n_u/n_s = 20$

Fig. 2: The figures show the percentage of undecided users for a typical power law distribution ($\alpha = 0.65$) with spread $\sigma = 2$ and $n_u = 1000$. Note that the undecided users decrease with time in all cases, but the convergence is faster when we use fewer but larger servers by setting n_u/n_s to be larger. Also, the smaller values of the look-ahead window τ result in faster convergence.



(a) $\alpha = 0.6$, $n_u/n_s = 1$ (b) $\alpha = 0.6$, $n_u/n_s = 20$

Fig. 3: Generally, as τ increases, convergence time increases but failure rate decreases. It is also true for larger servers (n_u/n_s = 20), only the failure has gone to zero for all investigated sliding window sizeτ.

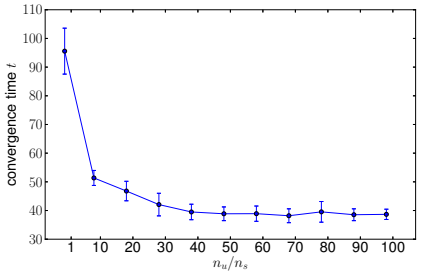


Fig. 4: As n_u/n_s increases fewer servers with larger capacity are used and convergence time decreases. The decrease is less pronounced beyond $n_u/n_s \geq 40$ under this setting ($\alpha = 0.65$, $\sigma = 2$, $\tau = 20$).

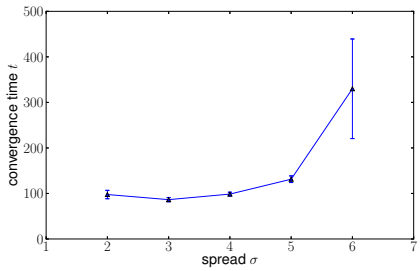


Fig. 5: There is a very small incremental benefit in using $\sigma = 3$ instead of 2, though higher values of $\sigma > 3$ only increased the convergence time. ($\alpha = 0.65, n_u/n_s = 1, \tau = 20, \kappa = 2.$)

contention in the network and is naturally defined as the ratio of the numbers of users in the system and total serving capacity that is available in the system. That is, $l \triangleq n_u/(\kappa \cdot n_s)$. For all three setting of Figure 2, we maintain a load $l = 0.5$. The figure shows that with fewer (but larger) servers (n_u/n_s is larger) the convergence time is faster, because having server capacity in a few larger servers provides a larger hit rate than having the same capacity in several smaller servers. Similar performance gains are also found in the context of web caching and parallel jobs scheduling [18]. Convergence times are plotted explicitly in Figure 4 for a greater range of user-to-server ratios. As n_u/n_s increases from 1 to 40, convergence time decreases. The decreases in convergence times are not significant beyond $n_u/n_s \geq 40$.

B. *Impact of sliding window* τ

The sliding window τ is the number of recent requests used by algorithm *GoWithTheWinner* to estimate the hit rate. As shown in Figure 3, there is a natural tradeoff between convergence time and failure rate. When τ increases, the users take longer to converge, as they require a 100% hit rate in a larger sliding window. However, waiting for a longer period also makes their decisions more robust. That is, a user is less

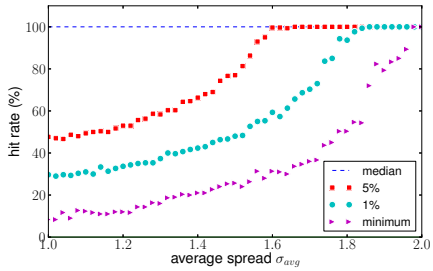


Fig. 6: Order statistics of the hit rate of the user population. ($\alpha = 0.65, n_u/n_s = 1, \tau = 10, \kappa = 2.$)

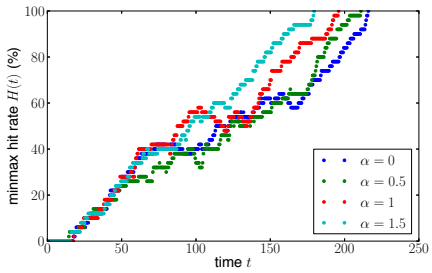


Fig. 7: Minmax hitrate versus time for different power law distributions.

likely to choose an overbooked server, since an overbooked server is less likely to provide a string of τ hits for large τ. In our simulations with many smaller caches ($n_u/n_s = 1$), when $\tau \leq 4$, users made quick choices based on a smaller sliding window. But, this resulted in the system converging to a non-optimal state 100% of the time. As τ further increases, failure rate decreases. The value of $\tau = 11$ is a suitable sweet spot as it results in the smallest convergence time for a zero failure rate. However, for fewer but larger servers ($n_s/n_u = 20$), all selections of window size τ (thus the small values like $\tau = 5$) yielded a 0% failure rate, while convergence time still increases as the window size gets larger.

C. Impact of spread σ

As shown in Theorems 3 and 4, a spread of $\sigma \geq 2$ is required for the system to converge to an optimal solution, while a spread of $\sigma = 1$ is insufficient. As predicted by our analysis, our simulations did not converge to an optimal state with $\sigma = 1$. Figure 5 shows the convergence time as a function of spread, for $\sigma \geq 2$.

As σ increases, there are two opposing factors that impact convergence time. The first factor is that as σ increases, each user has more choices and a user is more likely to find a suitable server with less load. On the other hand, an increase in σ also increases the total number of initial requests in the system that equals σn_u. Thus, the initiate server load increases in σ. These opposing forces result in a very small incremental benefit when using $\sigma = 3$ instead of 2, though the higher values of $\sigma > 3$ showed no benefit as convergence time increases with σ increases.

We established the "power of two random choices" phenomenon where two or more random server choices yield superior results to having just one. It is intriguing to ask *what percentage of users need two choices* to reap the benefits of multiple choices? Consider a mix of users, some with two random choices and others with just one. Let σ_{avg}, $1 \leq \sigma_{avg} \leq 2$, denote the average value of the spread among the users.

In Figure 6, we show different order statistics of the hit rate as a function of σ_{avg}. Specifically, we plot the minimum value, 1^{st}-percentile, 5^{th}- percentile and the median (50^{th}-percentile) of user hit rates after simulating the system for 200 time units. As our theory predicts, when $\sigma_{avg} = 2$, the minimum and all the order statistics converge to 100%, as all users converge to a 100% hit rate. Further, if we are interested in only the median user, any value of the spread is sufficient to guarantee that 50% of the users obtain a 100% hit rate. Perhaps the most interesting phenomena is that if $\sigma_{avg} = 1.7$, i.e., 70% of the users have two choices and the rest have one choice, the 5^{th}-percentile converges to 100%, i.e., all but 5% of the users experience a 100% hit rate. For a higher value of $\sigma_{avg} = 1.9$, the 1^{st}-percentile converges to 100%, i.e., all but the 1% of the users experience a 100% hit rate. This result shows that our algorithm still provides benefits even if only *some* users have multiple random choices of servers available to them.

D. Impact of demand distribution

We now study how hit rate changes with the exponent α in the power law distribution of Equation 4. Note that the distribution is uniform when $\alpha = 0$ and is the harmonic distribution when $\alpha = 1$. As α increases, since the tails fall as a power of α, the distribution becomes more skewed towards content items with a smaller rank. In Figure 7, we plot the minmax hitrate over time for different α, where we see that a larger α leads to faster convergence. The reason is that as the popularity distribution gets more skewed, a larger fraction of users will request the same popular content items, leading to higher hit rate and faster convergence. Thus, the uniform popularity distribution ($\alpha = 0$) is the worst case and the algorithm converges faster for the distributions that tend to occur more commonly in practice. Providing theoretical support for this empirical result by analyzing the convergence time to show faster convergence for larger α is a topic for future work.

V. Related work

Server selection algorithms have a rich history of both research and actual implementations over the past two decades. Several server selection algorithms have been proposed and empirically evaluated, including client-side algorithms that use historical performance feedback using probes [7], [5]. Server selection has also been studied in a variety of contexts, such as the web [5], [20], video streaming [21], and cloud services [22]. Our work is distinguished from the prior literature in that we theoretically model the "Go-With-The-Winner" paradigm

that is common to many proposed and implemented client-side server selection algorithms. Our work is the first formal study of the efficacy and convergence of such algorithms.

In terms of analytical techniques, our work is closely related to prior work on balls-into-bins games where the witness tree technique was first utilized [14]. Witness trees were subsequently used to analyze load balancing algorithms, and circuit-switching algorithms [4]. However, our setting involves additional complexity requiring novel analysis due to the fact that users can share a single cached copy of an object and the hitrate feedback is only a probabilistic indicator of server overload. Also, our work shows that the "power of two random choices" phenomenon applies in the context of content delivery, a phenomenon known to hold in other contexts such as balls-into-bins, load balancing [23], relay allocation for services like Skype [16], and circuit switching in interconnection networks [14].

VI. Conclusion

Our work constitutes the first formal study of the simple "Go-With-The-Winner" paradigm in the context of web and video content delivery. For web (resp., video) delivery, we proposed a simple algorithm where each user randomly chooses two or more candidate servers and selects the server that provided the best hit rate (resp., bitrate). We proved that the algorithm converges quickly to an optimal state where all users receive the best hit rate (resp., bitrate) and no server is overloaded, with high probability. While we make some assumptions to simplify the theoretical analysis, our simulations evaluate a broader setting that incorporates a range of values for τ and σ, varying content popularity distributions, differing load conditions, and situations where only some users have multiple server choices. Taken together, our work establishes that the simple "Go-With-The-Winner" paradigm can provide algorithms that converge quickly to an optimal solution, given a sufficient number of random choices and a sufficiently (but not perfectly) accurate performance feedback.

VII. Acknowledgements

This work was supported in part by NSF grant CNS-1413998. It was partially sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *INFOCOM, 2012 Proceedings IEEE*, pages 1620–1628. IEEE, 2012.

[2] Akamai. Akamai download manager. 2013. http://www.akamai.com/html/solutions/downloadmanager_overview.html.

[3] L. Breslau, P. Cue, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM*, 1999.

[4] R. Cole, B. M. Maggs, M. Mitzenmacher, A. W. Richa, K. Schröder, R. K. Sitaraman, B. Vöcking, et al. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 378–388. ACM, 1998.

[5] M. E. Crovella and R. L. Carter. Dynamic server selection in the internet. Technical report, Boston University Computer Science Department, 1995.

[6] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *Internet Computing, IEEE*, 6(5):50–58, 2002.

[7] S. G. Dykes, K. A. Robbins, and C. L. Jeffery. An empirical evaluation of client-side server selection algorithms. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1361–1370. IEEE, 2000.

[8] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for lru cache performance. In *Proceedings of the 24th International Teletraffic Congress*, ITC '12.

[9] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: A view from the edge. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07. ACM, 2007.

[10] P. Key, L. Massoulie, and P. Towsley. Path selection and multipath congestion control. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007.

[11] S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 211–224. ACM, 2012.

[12] C. Liu and P. Albitz. *DNS and Bind*. O'Reilly Media, Inc., 2009.

[13] C. Liu, R. K. Sitaraman, and D. Towsley. Go-with-the-winner: Client-side server selection for content delivery. *CoRR*, abs/1401.0209, 2014.

[14] M. Mitzenmacher, A. W. Richa, and R. Sitaraman. The power of two random choices: A survey of techniques and results. *COMBINATORIAL OPTIMIZATION-DORDRECHT-*, 9(1):255–304, 2001.

[15] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[16] H. X. Nguyen, D. R. Figueiredo, M. Grossglauser, and P. Thiran. Balanced relay allocation on heterogeneous unstructured overlays. In *INFOCOM*, 2008.

[17] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[18] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica. Sparrow: Distributed, low latency scheduling. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM.

[19] M. Raab and A. Steger. balls into binsa simple and tight analysis. In *Randomization and Approximation Techniques in Computer Science*, pages 159–170. Springer, 1998.

[20] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek. Selection algorithms for replicated web servers. *ACM SIGMETRICS Performance Evaluation Review*, 26(3):44–50, 1998.

[21] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao. Dissecting video server selection strategies in the youtube cdn. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 248–257. IEEE, 2011.

[22] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford. Donar: decentralized server selection for cloud services. In *ACM SIGCOMM Computer Communication Review*, volume 40. ACM, 2010.

[23] L. Ying, R. Srikant, and X. Kang. The power of slightly more than one sample in randomized load balancing. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015.

Utility-maximizing Server Selection

Truong Khoa Phan, David Griffin, Elisa Maini, Miguel Rio
University College London, UK
Email: {t.phan, d.griffin, e.maini, miguel.rio}@ucl.ac.uk

***Abstract*—This paper presents a new method for selection between replicated servers distributed over a wide area, allowing application and network providers to trade-off costs with quality-of-service for their users. First, we create a novel utility framework that factors in quality of service metrics. Then we design a polynomial optimization algorithm to allocate user service requests to servers based on the utility while satisfying transit cost constraint. We then describe an efficient - low overhead distributed model with the need to only know a small subset of the data required by a global optimization formulation. Extensive simulations show that our method is scalable and leads to higher user utility compared with mapping user requests to the closest service replica.**

I. Introduction and Motivation

As the Internet becomes the enabler for more types of services with a wider spectrum of requirements, pressure is being put onto the Internet ecosystem to facilitate service placement and to select the best replica for each user request at each instant in time. This replication always involves multi-stakeholder trade-offs involving costs (deployment and traffic related) and user quality of service (QoS).

There are many drivers for service replication, including server resilience, network diversity, and proximity of servers to users. Deploying services closer to the users allows the application providers to improve on QoS metrics like latency and/or throughput for all users. Some frameworks, like fog computing [1], even attempt to put service instances at the extreme edge of the network in locations such as access points.

Although in theory this replication could be optimal, in practice there are several obstacles: deployment costs vary between geographical areas and may be prohibitive in some locations, demand forecasting is inaccurate, flash crowds are unpredictable. Efficient allocation of user requests to service replicas will have to rely on a service selection at query time.

Service quality has two major sets of component metrics, relating to computation and networking parameters. Servers will have to be properly provisioned for the arrival rate and holding time of user requests otherwise users will be not served or blocked, increasing application latency. Network distance will have primarily an effect on end-to-end delay but, in many scenarios, causes an increase in packet loss and/or a decrease in good-put. The service selection system will have to take into account both computation and networking factors to optimize its selection.

Resolution involves converting a service name to a specific network locator for the selected replica. Our work assumes that the user's ISP is in the best position to make this selection. The ISP has accurate information regarding the user's position in the network, the current network status and, furthermore, it allows the ISP to apply traffic network policies in the selection process to reduce traffic costs. A centralized approach would, in theory, allow global optimization but it would often be unscalable and unrealistic. A central entity would not have access to information on the detailed user position, the network topology or current network status and would be incapable of implementing ISPs' specific traffic policies as it would have to arbitrate between conflicting policies of different ISPs which would be problematic from a business point of view. For those reasons, our server selection model can be implemented in a similar way of PaDIS [2] which allows ISPs to better assign users to servers by using their knowledge about network conditions and user locations.

In brief, the contributions of our work are as follows:

- Firstly, we introduce the *utility function* relating to one or more QoS metrics that allows application providers to define based on their application's requirements.
- Secondly, we design a *polynomial centralized optimization algorithm* that allows ISPs to redirect their users to the best replica, allowing to trade-off their traffic costs with users' QoS. In addition, the model allows to optimize for multi-services at the same time.
- Finally, we propose a *simple - efficient distributed model* that allows ISPs to run a local version of the selection algorithm without the need for global knowledge of all service replicas and network conditions.

This paper is organized as follows: Section II introduces the utility-maximizing server selection model and is followed by the optimization formulation in Section III. Section IV presents extensive evaluation. We finish by surveying related work in Section V and conclusions in Section VI.

II. Utility-maximizing Server Selection

The goal of utility-maximizing server selection is to provide the highest QoS for the greatest number of users. Our framework unifies the objectives of several stakeholders and the quality of service of the end users. In our approach, the stakeholders involved in service selection are as follows:

- Execution zones (EZ): These are the entities running the computational aspect of the distributed application. Typically they will be cloud/data centers but they can be smaller micro-data centers. They can be run by the ISPs themselves (current trends point to this being an

ISBN 978-3-901882-83-8

importance new revenue stream for them). Whatever the scenario, they want to maximize their revenue.

- Application (Service) Providers: These represent the organizations that wish to run distributed applications in execution zones. They may instantiate their replicas directly or use a third-party orchestrator (e.g. cloud broker). They will trade-off the deployment costs with the quality-of-service of their users. After deployment they publish the utility function of the service so that resolution algorithms can maximize this utility. It is noted that the utility function can simply be general (no sensitive) latency requirements for applications as shown in Fig. 2.
- Internet Service Provider: These will implement resolution algorithms to resolve users queries, mapping service names to locators. They will also trade-off the final quality of service of their users with the traffic costs imputed to them by these choices.

Application providers deploy service instances in execution zones. These replicas register with a local resolver in the ISP to which they are connected and send periodic updates. These messages contain for each service:

- The utility function of the service as described in section II-B.
- The number of available *session slots*. A session slot is a unit of measurement representing how many users can be accommodated simultaneously in a given service instance without blocking.
- Servicing execution statistics regarding the total demand on the service instance and the distribution of service duration times.

We define routing epoch as the interval between the resolution system making resolution decisions. Regarding to *session slots* announcement, if sessions are long compared to the routing epoch then the EZs simply announce a snapshot of what is available. However if session durations are short then an announcement of instantaneous availability is more-or-less meaningless. For example: assume a routing epoch of 10 seconds and a service S_1 with an average duration of 100 seconds and S_2 with an average duration of 1 second, and a single service instance for each service can each handle 2 sessions simultaneously. The EZ would announce 2 available session slots for S_1 as the current session is likely to last much longer than the routing epoch. However for S_2, if 2 available session slots are announced, it would mean that only two requests should be forwarded to that EZ, even though the currently active session (as well as those arriving in the near future) is very likely to end during the epoch. Therefore the number of session slot would be announced up to 20 for S_2, depending on the service arrival time.

Given those aforementioned stakeholders involving in the system, we present next the main criteria to do server selection.

A. Motivation Example

As an example in Fig. 1, assuming that there are two users requiring a voice service which is available in both

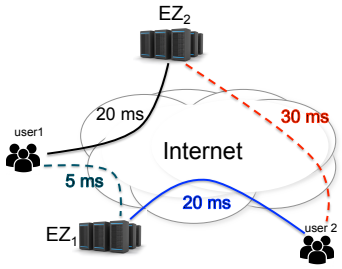


Fig. 1: Utility based vs. closest based selection

.

EZ_1 and EZ_2. However, each EZ can serve only one user at a time or they announce only one available session slot. Latencies between users and EZs are shown in Fig. 1. For the voice service, we observe that if bandwidth is enough, and when the latency is equal or less than 20 ms, humans' ears cannot distinguish between audio and real speech or in other words, people do not feel any disturbance [3]. Therefore, 5 ms or 20 ms latency gives the same (and the best) QoS for voice services. Note that we consider here the latency for on-going services, not including setup time. As shown in Fig. 1, the classical closest based selection algorithm would have solution: (user 1 - EZ_1) and (user 2 - EZ_2) (dash lines) as the minimum total latency is $(5{+}30) = 35$ ms. It means that user 1 can have the best QoS while user 2 sees some disruption in the voice quality. However, we see a better solution should be (user 1 - EZ_2) and (user 2 - EZ_1) in which both users get the best QoS with 20 ms latency. This is the motivation of our work to define a utility function applying in the server selection problem.

B. Utility Function

Our general utility function is grounded on practical research on quality of service utility [4], [5] and years of investigation on Mean Opinion Scores [6]. Our interval data points map to user ratings of *excellent*, *good*, *fair*, *poor* and *no service or blocked* (Fig. 2).

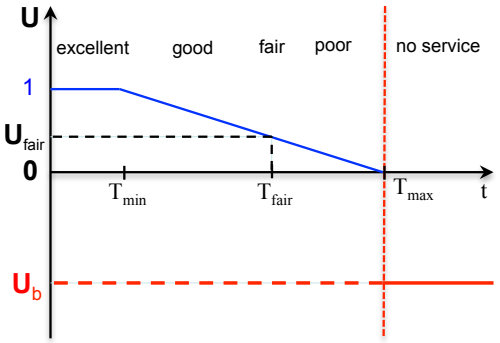


Fig. 2: Utility function vs. latency

.

In our utility framework, application providers determine utility function by the two latency thresholds: T_{min} and T_{max}.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Note that the utility is not restricted to only latency. In future work, we will extend the utility to be a combination of any QoS metrics such as latency, bandwidth, loss, etc. As shown in Fig. 2, we use a non-increasing piecewise linear utility function that is characterized by:

- If $t \leq T_{min}$: depending on the service type, an appropriate value of T_{min} is selected meaning that even if the latency reduces below this value, the improvement is not perceived by the users of that service, thus the utility is unchanged ($U_{max} = 1$). For instance, *voice over IP* requires $T_{min} = 20$ ms [3]; for *simple web services*, $T_{min} = 100$ ms gives users the feeling of instantaneous response [7].
- If $T_{min} < t \leq T_{max}$: QoS is within an acceptable range ($0 \leq U < 1$). User satisfaction reduces as the latency increases. We also define $T_{fair} \in [T_{min}, T_{max}]$ as the point from which users start to feel disappointed about the services as QoS is getting poor. Note that the value of T_{fair} is set depending on services and does not change the slope of the utility graph.
- If $T_{max} < t$: the request is *blocked* (*no service*) because the latency is beyond the acceptable range. Details on blocked requests are presented in Section III-B2.

Based on this utility function, the utility-maximizing solution for the problem in Fig. 1 will be (user 1 - EZ_2) and (user 2 - EZ_1) because both users will get the maximum utility with $t = T_{min} = 20$ms.

III. Server Selection Optimization

In this section, we present a mathematical formulation for the server selection problem that ISP's resolvers use to dynamically resolve names to locators. In general, the goal is to direct user requests to suitable execution zones (EZs) and satisfy a predefined objective function. We use the utility function defined in Section II-B to measure user satisfaction. We use linear programming to formulate the server selection problem which maximizes the total utility of all users while taking into account constraints on the data transit cost.

A central pre-requisite for our model is the existence of a forecasting demand component that provides an input to the optimization algorithm. Although client demand varies with time, work in the literature [8] points to reasonably stable demand within 10 minutes intervals. Note that, we aggregate individual users with the same preference to form a group. This can be done according to users' postal codes [8] or by users' IP prefixes [9]. Aggregation of this kind reduces the quantity of input variables for the optimization and also stabilizes request rates per-group [8].

A. Problem description

- Input: estimated user requests ($\mathcal{D}$); two threshold values (T^{ij}_{min} and T^{ij}_{max}) for each pair of (user group i, service j) defined in the utility function (Fig. 2); latency between user group i and EZ z for a specific service j is l^j_{iz}; unit data transit cost (c_{iz}); the maximum budget ($COST$) and available session slots at each EZ (S^j_z).
- Objective: maximize the performance (total utility) of users for multi-services j while considering the trade-off between the performance and the data transit cost.
- Output: $x^j_{iz} \in [0, 1]$: fraction of user group i connecting to EZ z for service j.

We define a utility function as described in Section II-B as follows:

$$u_{ij} = \begin{cases} 1 & \text{if } t_{ij} \leq T^{ij}_{min} \\ \frac{-t_{ij}+T^{ij}_{max}}{(T^{ij}_{max}-T^{ij}_{min})} & \text{if } T^{ij}_{min} < t \leq T^{ij}_{max} \\ U_b & \text{otherwise} \end{cases}$$

The utility is defined for each pair of (user group i, service j). When the latency is larger than T^{ij}_{max}, the request is considered to be blocked. We set U_b to be a small negative value to indicate the utility of a blocked request. More details on how to set value for U_b are presented in Section III-B2. We first present a centralized model and then introduce a distributed one which is more suitable for Internet-scale deployment.

B. Centralized model

Given the key notations in Table I, we use linear programming to formulate the utility-maximizing server selection problem.

1) Linear Program Formulation:

$$\max[\sum_{(i,j)\in\mathcal{D}} u_{ij}] \quad (1)$$

s.t.

$$\sum_{z\in\mathcal{Z}} x^j_{iz} = 1 \qquad \forall (i,j) \in \mathcal{D} \quad (2)$$

$$\sum_{i\in\mathcal{I}} d_{ij} x^j_{iz} \leq S^j_z \qquad \forall j \in \mathcal{J}, z \in \mathcal{Z} \quad (3)$$

$$t_{ij} = \sum_{z\in\mathcal{Z}} l^j_{iz} x^j_{iz} \qquad \forall (i,j) \in \mathcal{D} \quad (4)$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{D} \quad (5)$$

$$y_{ij} \geq t_{ij} - T^{ij}_{min} \qquad \forall (i,j) \in \mathcal{D} \quad (6)$$

$$u_{ij} = \frac{T^{ij}_{max} - T^{ij}_{min} - y_{ij}}{T^{ij}_{max} - T^{ij}_{min}} \qquad \forall (i,j) \in \mathcal{D} \quad (7)$$

$$\sum_{z\in\mathcal{Z}} \sum_{(i,j)\in\mathcal{D}} c_{iz} b_{ij} x^j_{iz} \leq COST \quad (8)$$

$$x^j_{iz} \in [0,1], u_{ij} \leq 1 \qquad \forall (i,j) \in \mathcal{D}, z \in \mathcal{Z} \quad (9)$$

Explanation:

- The objective function (1) maximizes the total utility over all user groups.
- Constraint (2): all the requests of user group i for a specific service j have to be served.
- Constraint (3): each EZ has a limited capacity (bandwidth and computational resources) for deploying instances of a specific service type. It may be the case that some special services (e.g. those that require GPU processing) can only be deployed at certain EZs with the necessary hardware

TABLE I: Key Notations (in Alphabetical Order)

b_{ij}	bandwidth required by user i to get service j
$COST$	the maximum transit cost (budget)
c_{iz}	unit transit cost between user i and EZ z
$\mathcal{D}$	set of user requests $\mathcal{D} = \{(i,j), \forall i \in \mathcal{I}, j \in \mathcal{J}\}$
d_{ij}	requested session slot of user i to service j
$\mathcal{I}$	set of user groups $\mathcal{I} = \{i\}$
$\mathcal{J}$	set of services $\mathcal{J} = \{j\}$
l_{iz}^j	latency between user i and EZ z for service j
S_z^j	available session slot of service j at EZ z
t_{ij}	average latency of user i to get service j
$\mathcal{Z}$	set of execution zones (EZ) $\mathcal{Z} = \{z\}$
U_b	utility value of a blocked user
u_{ij}	utility of user i when getting service j
x_{iz}^j	fraction of user i connects to EZ z to get j
y_{ij}	variable used to compute the utility

available. This constraint ensures that the number of session slots available in an EZ is sufficient to serve user requests.

- Equation (4) is used to compute the average latency for the user group i to get the service j. We model connectivity as a full mesh between user groups and EZs. However, for the input of the LP, we do not consider any connections between user group i and EZ z to get service j if the latency $l_{iz}^j > T_{max}^{ij}$. This step guarantees that, even without adding explicit constraints in the model, the latency for any user group i to connect to any EZ z to get service j is less or equal to T_{max}^{ij}.
- Constraint (5) - (6) ensure that $y_{ij} \geq 0$ if $t_{ij} \leq T_{min}^{ij}$, otherwise $y_{ij} \geq t_{ij} - T_{min}^{ij} > 0$.
- Equation (7) is used to model the utility function defined in Section III-A. There are two possibilities:
  - If $t_{ij} \leq T_{min}^{ij}$, based on constraints (5) - (6), y_{ij} can be any value greater or equal to 0, however, due to the objective function maximizing utility (7) the formulation will choose a minimum value of y_{ij}, or in other words, y_{ij} is set to 0 and thus, $u_{ij} = 1$.
  - Similarly, if $t_{ij} > T_{min}^{ij}$, the formulation will choose $y_{ij} = t_{ij} - T_{min}^{ij}$ and thus $u_{ij} = \frac{-t_{ij}+T_{max}^{ij}}{T_{max}^{ij}-T_{min}^{ij}}$.
- Constraint (8) limits the data transit cost. As shown in [9], [10], the linear transit cost we use here is also a good approximation for the 95-th percentile transit cost.

An important remark is that our formulation allows to optimize for multi-services at the same time, i.e. maximizing the total utility for all pairs of (user i, service j). For instance, assume that we have video streaming and voice services, each potentially has different QoS requirements. If we optimize for video streaming first, then the remaining resources are dedicated for the voice service and vice versa. This will end up with a sub-optimal solution. Our model allows to optimize for multi-services at the same time, therefore it guarantees to find a global optimal solution.

The optimization formulation above is a (pure) linear programming model one; hence it can be solved efficiently in polynomial time. The number of variables x_{iz}^j in the LP problem is $|\mathcal{I}| \times |\mathcal{Z}| \times |\mathcal{J}|$ where $|\mathcal{I}|$ is the number of user groups, $|\mathcal{Z}|$ is the number of EZs and $|\mathcal{J}|$ is the number of service types. Since $|\mathcal{Z}|$ and $|\mathcal{J}|$ are usually much smaller than $|\mathcal{I}|$, the worst case complexity of the LP problem will be $O(|\mathcal{I}|^{3.5})$ [9]. We report the execution time of the algorithm in Section IV-C1.

2) Blocked user requests: When there are not enough resources (session slot or cost) the constraints (3) and (8) in the LP are violated and the LP ends up with no solution. We address this by allowing user requests to be *blocked* when there are insufficient resources. To model this, we define a virtual EZ with very large capacity so that the constraint (3) cannot be violated. The transit cost between users and the virtual EZ is zero. The latency between all users to this virtual EZ is set at a value which is larger than T_{max}, therefore the utility for a blocked request is $U_b < 0$ (Fig. 2). We evaluate different values of U_b and show its impact on the number of requests to be blocked when running the optimization model (Section IV-A2). Intuitively, the closer to 0 the value of U_b is, the more possibility for requests to be blocked because there is a reduced difference in utility between a request at T_{max} ($U_{T_{max}} = 0$) and a blocked one (U_b). It is noted that, with the virtual EZ, the LP always finds a feasible solution because the constraints (3) and (8) cannot be violated, but the total utility could be extremely (negative) small. And those requests that have to go to the virtual EZ are considered to be blocked.

C. Distributed model

Although our centralized optimization model can be solved in polynomial time, it is impractical in real deployments as a single global resolver would be required to collect information from all EZs and networks and also to handle resolution requests from all users. Moreover we want to put the ISP (*resolver*) at the center of decision making in order for local traffic policies to be applied.

Designing an efficient distributed algorithm is a classical problem [8], [10], [11], and it would satisfy the following general requirements:

- *(1) Low overhead:* small number of control messages exchanged. In addition, it should guarantee a fair share based on demand of each resolver.
- *(2) Convergence:* the algorithm is guaranteed to be always converged to a stable solution.
- *(3) Efficiency:* solution of the distributed algorithm is close to the centralized one.

Existing work in literature can satisfy the requirements (2) and (3) by using optimization decomposition methods [8], subgradient methods [11] or alternating direction method of multipliers [10]. However, they end up with high complexity formulation and require high control overhead. Potentially, control messages can be exchanged between (in both directions): resolvers - resolvers, resolvers - EZs, and EZs - EZs. In this work, we propose a novel distributed model satisfying all the three aforementioned requirements. Compared with existing work, our model is simpler (still can be solved

TABLE II: Key Notations in Distributed Algorithm

$A_i^z(k)$	allocated session slots at z by R_i in epoch k
C^z	total session slots at EZ z
k	epoch (iteration) number
M	number of resolvers that share an EZ
N	number of EZs that one resolver can see
R_i	resolver i
$S_i^z(k)$	available session slots at z seen by R_i in epoch k
Z	set of execution zones

in polynomial time) and low overhead in which only one-way control messages from EZs to resolvers are needed. In addition, the messages exchanged are simple as we describe later.

1) Distributed algorithm: We divide the time into **intervals** in which we assume the traffic demand is unchanged (e.g. 10 minutes as observed in [8]). Each interval is sub-divided into **epochs** and the distributed algorithm is run at the beginning of each epoch. We call *visibility set* be a subset of EZs that are closest to a resolver and can be seen by that resolver.

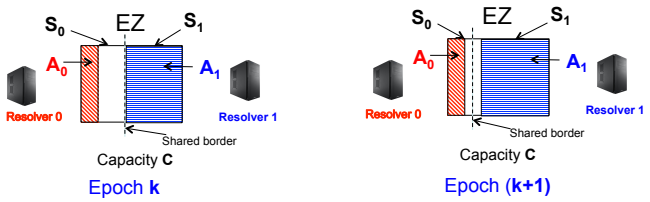


Fig. 3: EZ z is shared by two resolvers R_0 and R_1

.

We introduce some notations used in the distributed algorithm (Table II). Considering an EZ z with total available session slots C^z which is shared by M resolvers. At an epoch $k \geq 0$, let a resolver $R_i (0 \leq i \leq M-1)$ see $S_i^z(k) \leq C^z$ session slots from the shared EZ. To guarantee capacity constraint, we have $\sum_{i=0}^{M-1} S_i^z(k) \leq C^z$. Let $A_i^z(k) \leq S_i^z(k)$ be the number of session slots that the resolver R_i allocates for its users to connect to the EZ z at the epoch k. A visualization of those notations are shown in Fig. 3. The algorithm, step-by-step, at each resolver is as follows:

1) *At the beginning of each interval*: collect the latest estimated local user requests and network metrics (e.g. latency between users and EZs).
2) *At the beginning of each epoch*: each EZ announces the latest capacity (C^z) and the total-in-allocation (total-in-use) session slots ($\sum_{i=0}^{M-1} A_i^z(k)$) at that EZ to resolvers.
3) Each resolver updates available session slots that it can use in the next epoch as follows:

$$S_i^z(k+1) = A_i^z(k)\big[1 + \frac{C^z - \sum_{i=0}^{M-1} A_i^z(k)}{\sum_{i=0}^{M-1} A_i^z(k)}\big] \quad (10)$$

if $\sum_{i=0}^{M-1} A_i^z(k) = 0$, we set $S_i^z(k+1) = C^z$.
4) Given new available session slots from EZs, resolvers execute the linear program in Section III-B to find which users should connect to which EZs to get services.

By using the equation (10), we show that the distributed algorithm satisfies all the requirements mentioned in III-C:

- Each resolver requires local user demand and the session slots have been used in the previous epoch ($A_i^z(k)$). In addition, each resolver uses the LP in III-B, thus the distributed algorithm can be solved in polynomial time.
- Low overhead: only one-way message from EZs to resolvers to update information about total capacity (C^z) and total in-use session slots at the previous epoch ($\sum_{i=0}^{M-1} A_i^z(k)$). In addition, we show that the equation (10) also achieves the *fair share on demand* requirement. As shown in Fig. 3, at epoch k, the resolver R_0 just uses a small fraction of its shared available session slots ($A_0 < S_0$) while R_1 requires all the slots that it can see ($A_1 = S_1$). Therefore, in the epoch $(k+1)$, we should move the shared border to the left (but do not touch the red area - the allocated slots of R_0) so that there will be more free space for R_1 to forward its requests to the EZ if needed. This can be done automatically by using the equation (10) (see the example in III-C2).
- We show, both by mathematical proof and simulations that local decisions always converge within a handful of iterations and the solution of distributed algorithm is close to the centralized one. However, because of limited space, we omit the mathematical proof in this paper and will present it in a journal version.

Initially, when services are first deployed, each EZ announces its available session slots to all resolvers that can see it. Given the available session slots and the local user demand, each resolver executes the linear formulation in section III-B to find a solution for its users. In this initial step, EZs can be overloaded as they are shared by many resolvers, but there is no message between resolvers to say that. However, by using the equation (10) to update available capacity at EZs after each epoch, the capacity constraints are not violated after the initial step. We present a simple example to make the algorithm clear.

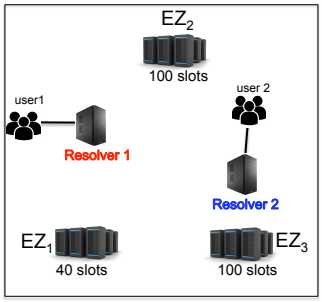


Fig. 4: Example of distributed algorithm

.

2) Examples of distributed algorithm: Assume that user 1 requires 100 slots and user 2 requires 80 slots. Capacities of EZs are shown in Fig. 4. The latencies between resolvers, users and EZs are as follows:

- $l(R_1, EZ_2) < l(R_1, EZ_1) < l(R_1, EZ_3)$
- $l(R_2, EZ_3) < l(R_2, EZ_2) < l(R_2, EZ_1)$

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

- $T_{min} < l(usr_1, EZ_2) < l(usr_1, EZ_1) < l(usr_1, EZ_3)$
- $T_{min} < l(usr_2, EZ_2) < l(usr_2, EZ_3) < l(usr_2, EZ_1)$

Recall that depending on the size of the *visibility set*, we can have different solutions for the server selection problem. Using the above network metrics, we consider the example with two scenarios:

- Scenario 1 (*visibility set size is* 1): the resolver 1 can only see EZ_2 (as EZ_2 is the closest EZ of R_1) and the resolver 2 can only see EZ_3. Therefore, the solution will be: the resolver 1 sends all 100 requests to EZ_2 and similarly, all requests of the resolver 2 go to EZ_3. This solution does not change if the user requests are unchanged between epochs.

- Scenario 2 (*visibility set size is* 2): resolver 1 can see (EZ_2 and EZ_1) and resolver 2 can see (EZ_3, EZ_2). Assume that the requests do not change, we present results for each resolver within 2 epochs (or 2 iterations of the distributed algorithm).

- Epoch 0:
  - Resolver 1 sees from EZ_1: $S_1^1(0) = 40$, and from EZ_2: $S_1^2(0) = 100$. As $l(usr_1, EZ_2) < l(usr_1, EZ_1)$, it forwards all 100 requests to EZ_2.
  - Resolver 2 sees from EZ_2: $S_2^2(0) = 100$, and from EZ_3: $S_2^3(0) = 100$. As $l(usr_2, EZ_2) < l(usr_2, EZ_3)$, it assigns all 80 requests to EZ_2.

  The total allocated session slots at EZ_2 is 180, and the EZ_2 is overloaded at epoch 0.
- Epoch 1:
  - Resolver 1 updates available session slots using the equation (10):
    - $S_1^1(1) = C^1 = 40$ (as $A_1^1(0) + A_2^1(0) = 0$)
    - $S_1^2(1) = 100 \times (1 + \frac{100-180}{180}) = 55$

    Solution after epoch 1 is: 40 slots go to EZ_1; 55 slots go to EZ_2 and 5 slots are blocked (as there are insufficient session slots).

  - Resolver 2 updates available session slots using the equation (10):
    - $S_2^3(1) = C^3 = 100$ (as $A_1^3(0) + A_2^3(0) = 0$)
    - $S_2^2(1) = 80 \times (1 + \frac{100-180}{180}) = 44$

    Solution after epoch 2 is: 44 slots go to EZ_2; 36 slots go to EZ_3 and no slots are blocked.

It is clear that, after epoch 1, thanks to the equation (10), no EZ is overloaded. In this example, the solution does not change after 2 epochs as long as the user demands do not change. It means that the distributed algorithm converges to a stable solution. On the other hand, session slots are assigned proportionally to the requirement of each resolver. For instance, in the stable solution, R_1 and R_2 respectively use 55 and 44 slots from the EZ_2. It is because in epoch 0, R_1 requires 100 slots while R_2 needs only 80 slots ($\frac{100}{80} \simeq \frac{55}{44}$). We call this as *fair share on demand*.

IV. Simulation Results

We solve the linear program model using IBM CPLEX solver [12]. All computations were carried out on a computer equipped with a 3 GHz CPU and 8 GB RAM. Our evaluation is through simulation and consists of 5 main parts. Firstly, we evaluate the algorithms with different parameters: $\frac{supply}{demand}$ ratios, U_b on blocking probability and visibility set sizes for the distributed algorithm. Next, we compare our novel utility-maximizing server selection (USS) with the classical closest based server selection algorithm. Then, we evaluate the distributed algorithm and compare with the centralized one. Next, we show the impacts of mismatch between supply and demand on the server selection solution. And lastly, we discuss on the inaccuracies of demand forecasting when using our algorithm.

We use a dataset with 2508 data centers distributed in 656 cities all over the world [13]. For the distributed model, we assume that each city has one resolver. Since data centers in a city are geographically close to each others, we group them as one execution zone (EZ). The capacity of an EZ is proportional to the number of data centers in that city. We assume that the services are available in all EZs. The data transit cost is based on the Amazon EC2 charging model. The user demand is modeled as Poisson process and is proportional to the population of each city [14]. The latency between users and execution zones are collected based on Haversine distance, the shortest distance between two points around the planet's surface [15].

A. Different Parameters for the Algorithm

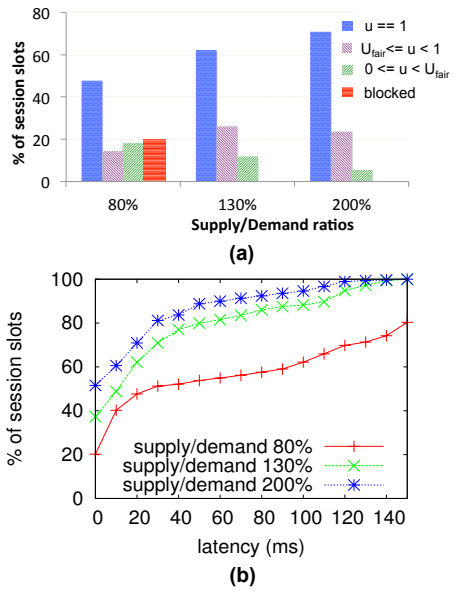


Fig. 5: Different $\frac{supply}{demand}$ ratios

.

1) Different Supply/Demand Ratios: We first find server selection solutions for different $\frac{supply}{demand}$ ratios with the centralized algorithm. We set $T_{min} = 20$ ms, $T_{fair} = 100$ ms and $T_{max} = 150$ ms for all pairs of (group user, service). In Fig. 5, we show the utility and the Cumulative Distribution Function (CDF) of latency for three scenarios of $\frac{supply}{demand}$ ratio: 80%, 130% and 200%. $\frac{supply}{demand} = 80\%$ means that the total

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

available capacity in all EZs is scaled down to equal to 80% of the total requests. As a result, 20% of the requests will be blocked while maximizing total utility of the served requests. In the CDF of latency in the scenario 80% (Fig. 5b), only 80% of requests receive service with less than $T_{max} = 150$ and the remaining requests are blocked. For the two other scenarios (130% and 200%), since there are sufficient session slots, no user request is blocked. Obviously, the greater the supply of session slots, the better solution we get in terms of utility and latency (Fig. 5).

2) Utility of a Blocked User: As shown in Section III-B2, our algorithm allows to block user requests while maximizing the total utility. By selecting different values of utility for a blocked request ($U_b < 0$), we obtain solutions with different blocking probabilities. We show in Fig. 6 the results for the *centralized algorithm* with different values of U_b. When U_b

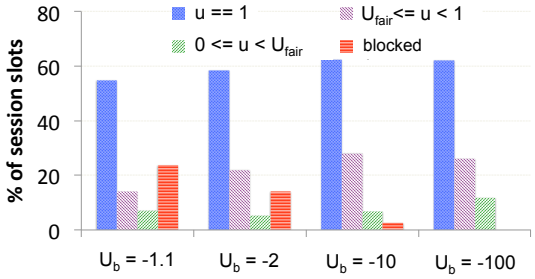


Fig. 6: Different values of U_b

.

is close to 0, e.g. $U_b = -1.1$ or $U_b = -2$, blocking user requests does not incur much penalty in the total utility. A significant number of requests are blocked despite total utility being maximized. When U_b is much more smaller (e.g. $U_b = -100$), blocking a single request can dramatically reduce the total utility, thus the algorithm tries to avoid as many requests being blocked as possible. In Fig. 6, when $U_b = -100$, no user request is blocked.

In the remaining evaluation, if not stated otherwise, default values are used as follows: $\frac{supply}{demand}$ ratio = 130%, $U_b = -100$, $T_{min} = 20$ ms, $T_{fair} = 100$ ms and $T_{max} = 150$ ms.

3) Different visibility sets: In a distributed manner, each resolver only sees its local user demand and a subset of EZs which is called *visibility set*. We vary the size of the visibility set by changing the parameter N, the percentage of the total 656 execution zones that can be seen by a resolver. For example, $N = 0.3\%$ means that each resolver can see its 2 closest EZs.

Intuitively, when N increases, more session slots are available for a resolver to allocate user requests. As shown in Fig. 7, the percentage of blocked requests reduces as we increase N. However, as resolvers greedily allocate their user requests to the best EZs in their visibility set, users in "poor resource" areas do not have enough session slots and still we can see a few of users are blocked even $N = 100\%$. Note that in the centralized algorithm, there is no blocked user request when the $\frac{supply}{demand}$ ratio is 130%.

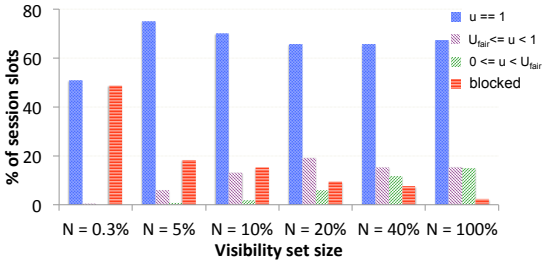


Fig. 7: Utility with different visibility set sizes

.

B. USS vs. closest selection algorithm

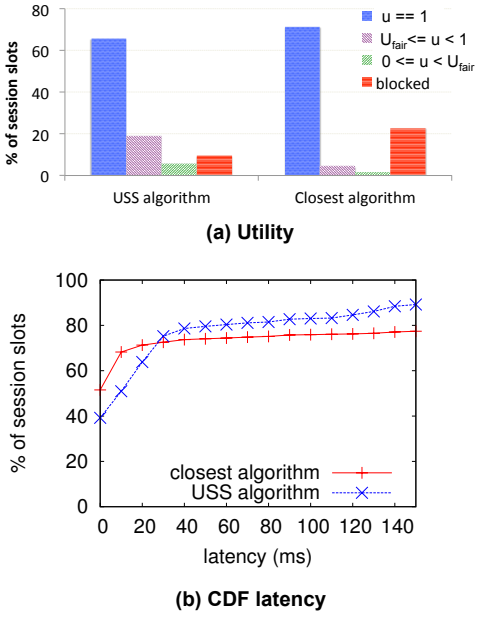


Fig. 8: Voice: USS vs. closest algorithm

.

Given the parameters in IV-A2, Fig. 8 shows a comparison between our utility-maximizing server selection (USS) and the classical closest algorithm with $N = 20\%$. The closest algorithm tries to allocate user requests to nearby EZs that have available session slots. If the closest EZ does not have available session slots, the algorithm considers the next closest one and so on. Only the case there is no available session slot within the latency range of T_{max}, requests have to be blocked. After finding the latency for user requests in the closest solutions, we compute the utility corresponding to the voice ($T_{min} = 20$ ms, $T_{fair} = 100$ ms and $T_{max} = 150$ ms [16]) (Fig. 8a). We can see the USS algorithm performs better with less blocking probability. This is because the USS algorithm is less greedy, providing more flexibility for requests to connect to many servers which have latency less than T_{min}. Taking a closer look at the CDF of latency (Fig. 8b), more requests get low latency in the closest algorithm, however more requests are also blocked due to its greedy behavior.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

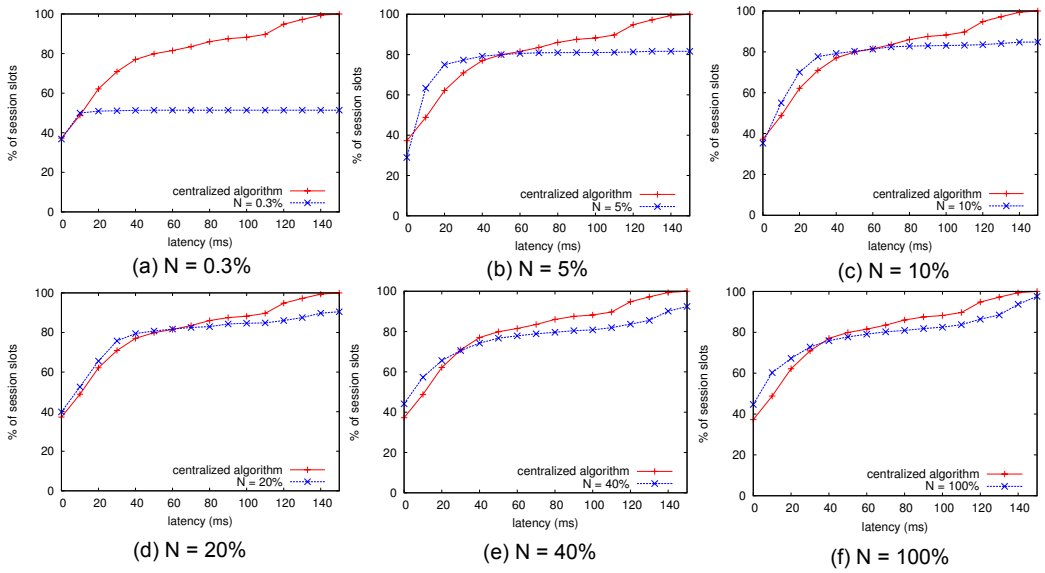


Fig. 9: Latency with different visibility set sizes

.

C. Distributed Algorithm

1) Distributed vs. centralized algorithm: We show in Fig. 9 the CDF of latency for the centralized and the distributed algorithms with different visibility set sizes. One of the goal when designing the distributed algorithm is that it should perform well, close to the centralized one. However, performance of the distributed algorithm depends on how much resource a resolver can see. As shown in Fig. 9, as visibility set size N increases, each resolver can see more available session slots, therefore less user requests are blocked. To report on execution time, the centralized algorithm with full knowledge of execution zones and user demands takes about 2 minutes to find an optimal solution, while the distributed algorithm only requires a few seconds to finish.

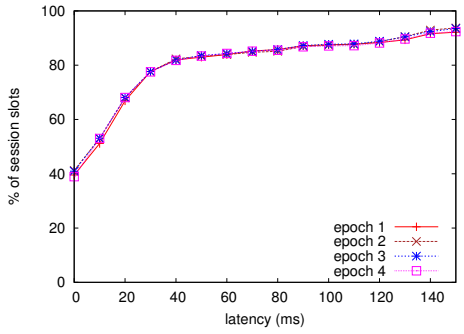


Fig. 10: Convergence of distributed algorithm

.

2) Convergence of distributed algorithm: To evaluate the convergence of the *distributed algorithm*, Fig. 10 shows the quality of the solution after 4 epochs in case visibility set size $N = 20\%$. We see that the four lines in Fig. 10 are nearly overlapped. After the first epoch, we already have a reasonably good solution and is close to the final solution. This result confirms a fast convergence of the distributed algorithm.

D. Mismatch between supply and demand

To evaluate the impact of mismatch between supply and demand, we first run the *centralized model* (section III) but without the capacity and the cost constraints. That is to find how many session slots are needed at every EZ for an optimal server selection solution. Then we scale these values to achieve 130% $\frac{supply}{demand}$ ratio. We call this configuration be the *perfect allocation*. Next, we create different levels of mismatch between supply and demand by varying a parameter "$X\%$ rand." (Fig. 11). This means that for each EZ, we remove $X\%$ of its session slots from the perfect allocation configuration. Then, we mix the removed session slots of all EZs and scatter them uniformly to all EZs. This guarantees that the total session slots of EZs in all cases (perfect allocation and "$X\%$ rand.") are the same. "0% rand." is equivalent to the perfect allocation while in "100% rand.", there is a uniform distribution of sessions slots between all EZs. Fig. 11 shows

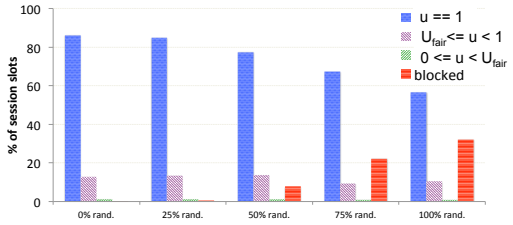


Fig. 11: Utility with different mismatch levels

evaluation results for the *distributed algorithm* with visibility set size $N = 5\%$ with different value of "$X\%$ rand.". With the perfect allocation "0% rand.", the distributed algorithm

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

performs well with no blocked requests. It is clear that by increasing $X\%$, more requests are blocked as we increase the level of mismatch between local supply and demand. It is noted that because we use a 130% $\frac{supply}{demand}$ ratio, the scenario "25% rand." is within an acceptable range of mismatch and the solution is close to the "0% rand." case.

E. Impact of inaccuracy in demand forecast

A central prerequisite for our model is the existence of a forecasting demand component that provides an input to the optimization algorithm. We discuss in this section the robustness of our solution vs. inaccuracy in forecasting demand. As shown in Fig. 11 (distributed algorithm with $N = 5\%$), the cases "50% rand." and "75% rand." respectively have around 8% and 22% of blocked user requests. On the other hand, in Fig. 7, with $N = 5\%$, there are around 18% of requests are blocked. This would mean that our results for the distributed algorithm in this paper (except Fig. 11) is corresponding to $50\% - 75\%$ inaccuracy in the forecasting demand compared to the perfect allocation case. Therefore, this mismatch leads to worse solutions. However, as shown in Fig. 7, we still can find good solution (small fraction of blocked users) for the distributed algorithm if the visibility set is large enough, for instance $N \geq 20\%$.

V. Related Work

Server selection: our work is closely related to recent work on optimizing performance-cost for server selection [8], [9]. For example, Wendell et al. [8] introduce DONAR - a decentralized replica-selection system that considers client locality, server load, and policy preferences. Like DONAR, our model can perform balancing client requests across replicas by manually setting capacity cap at each execution zone. Zhang et al. [9] focus on optimizing cost and performance in online service provider networks. The objective is to search for the optimal "sweet-spot" in the performance-cost Pareto front. Auspice [17] uses a heuristic placement algorithm to determine the locations of active replicas so as to minimize client-perceived latency. In general, these works use the classical closest method saying that the closer the servers are, the better QoS users can perceive. Our study can be a complementary for previous work as we define utility, a more general framework to qualify QoS compared to the classical closest approach. In addition, our polynomial algorithm can perform optimization for multi-services at the same time.

Network latency and traffic demand estimation: recent works have shown that the IP geolocation of the user provides accurate and predictable network latency [18]. This has been confirmed not only by third-party datasets such as Peerwise [19] and iPlane [20], but also by our own extensive active measurements [21], [22]. On the other hand, work in literature shows that client request rate can be sufficiently predictable under short interval (e.g. 10 minutes [8]). These works are useful as they provide accurate inputs for our optimization model.

VI. Conclusions and Future Work

This paper presented *utility-maximizing server selection*, a novel method to implement service instance selection that allows for trading-off user QoS with traffic cost. Compared with the classical closest approach, our utility framework allows reducing blocking probability while maintaining good utility for users. As further work, we are working on the modeling of incentives between application and network providers and how to address the scenarios where the ideal choice of server is not the same for both stakeholders. In addition, we are planning to extend the utility function to support more QoS metrics.

Acknowledgment

This research has received funding from the Seventh Framework Programme (FP7/2007-2013) of the European Union, through the FUSION (grant agreement 318205) projects.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and its Role in the Internet of Things," in *MCC*, 2012.
[2] I. Poese, G. Smaragdakis, B. Frank, S. Uhlig, B. Ager, and A. Feldmann, "Improving Content Delivery with PaDIS," *IEEE Internet Computing*, vol. 16, no. 3, pp. 46–52, 2011.
[3] M. Stone and B. Moore, "Tolerable Hearing Aid Delays. Est. of Limits Imposed by the Auditory Path Alone using Simulated Hearing Losses," *Ear and Hearing*, vol. 20, no. 3, 1999.
[4] M. A. Khan and U. Toseef, "User Utility Function as Quality of Experience (QoE)," in *ICN*, 2011.
[5] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguade, "Utility-based Placement of Dynamic Web Applications with Fairness Goals," in *NOMS*, 2008.
[6] "Recommendation p.800 (08/96)," http://www.itu.int/rec/T-REC-P.800-199608-I/en.
[7] J. Nielsen, "Usability Engineering: Response Times: The Three Important Limits," 1993.
[8] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford., "DONAR: Decentralized Server Selection for Cloud Services," in *SIGCOMM*, 2010.
[9] Z. Zhang, Y. Hu, M.Zhang, R.Mahajan, A. Greeberg, and B. Christian, "Optimizing Cost and Performance Online Service Provider Networks," in *NSDI*, 2010.
[10] H. Xu and B. Li, "Joint Request Mapping and Response Routing for Geo-distributed Cloud Services," in *INFOCOM*, 2013.
[11] S. Boyd and A. Mutapcic, "Subgradient Methods," in *Lecture notes of EE364b, Stanford University*, 2006.
[12] Www-01.ibm.com/software/commerce/optimization/cplex-optimizer.
[13] Http://www.datacentermap.com/.
[14] Https://github.com/richardclegg/multiuservideostream.
[15] G. V. Brummelen, *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton Uni. Press, 2013.
[16] S. Gangam, J. Chandrashekar, I. Cunha, and J. Kurose, "Estimating TCP Latency Approximately with Passive Measurements," in *PAM*, 2013.
[17] A. Sharma, X. Tie, D. Westbrook, H. Uppal, A. Yadav, and A. Venkataramani, "A Global Name Service for a Highly Mobile Internetwork," in *SIGCOMM*, 2014.
[18] S. Agarwal and J. Lorch, "Matchmaking for Online Games and Other Latency-sensitive P2P Systems," in *SIGCOMM*, 2009.
[19] M. Lu, J. Wu, K. Peng, P. Huang, J. Yao, and H. Chen, "Design and Evaluation of a P2P IPTV System for Heterogeneous Networks," *IEEE ToM*, vol. 9, no. 8, pp. 1568–1579, 2007.
[20] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, , and A. Venkataramani, "iPlane: An information Plane for Distributed Services," in *NSDI*, 2006.
[21] R. Landa, J. T. Araujo, R. G. Clegg, E. Mykoniati, D. Griffin, and M. Rio, "The Large Scale Geography of Internet Round Trip Times," in *IFIP Networking*, 2013.
[22] ——, "Measuring the Relationships between Internet Geography and RTT," in *ICCCN*, 2013.

TCP Hollywood: An Unordered, Time-Lined, TCP for Networked Multimedia Applications

Stephen McQuistin
University of Glasgow, UK
sm@smcquistin.uk

Colin Perkins
University of Glasgow, UK
csp@csperkins.org

Marwan Fayed
University of Stirling, UK
mmf@cs.stir.ac.uk

***Abstract*—Ossification of the transport-layer limits networked multimedia applications to use TCP or UDP, despite standardisation of new transport protocols that better support their requirements. To improve transport for these applications, we present TCP Hollywood, an unordered, time-lined, TCP variant designed to support real-time multimedia traffic while being widely deployable. Analysis of the protocol indicates that it increases the utility of the network in lossy conditions where total one-way delay is constrained, such as with telephony applications and low-latency video streaming. This allows retransmissions to be useful in cases where they are not with standard TCP, improving the timely good-put of the protocol and reducing overheads. Initial experiments show that TCP Hollywood is deployable on the Internet, successfully operating on all major fixed and mobile networks in the UK, with safe failure modes.**

I. Introduction

Real-time networked multimedia applications have long contributed to Internet traffic. This can take the form of telephony [1], video conferencing [2], live or on-demand TV and movies [3], [4], or user-generated video. These applications, and the traffic they generate, are rapidly increasing in popularity, and now comprise the majority of Internet traffic [5].

The nature of such real-time traffic is that it prefers predictable and bounded latency to strict reliability, since data that arrives too late is as bad as data that does not arrive at all. This suggests that data should be sent in packets that can be independently decoded [6], to allow them to be processed irrespective of the loss or delay of other packets. However, the requirement for efficient media compression leads to interdependence between packet contents and codecs that operate across multiple frames. When coupled with challenging network environments, such as mobile wireless, that have unreliable delivery and unpredictable latency, the requirements for effective media transport become difficult to satisfy.

Applications access the network via the transport layer. The transport protocol should provide services to meet the application demands, abstracting away details of the transport process, and delivering data with an appropriate degree of reliability and timeliness. For real-time networked multimedia, the transport should be trusted to minimize transport-induced delay, and should respect (partial) reliability semantics pertaining to media importance, deadlines, and dependencies.

Message-oriented transports, such as SCTP [7] and DCCP [8], ought to be suitable building blocks, but their deployment is restricted by NATs, firewalls, and other middleboxes [9]. This leaves real-time applications to use UDP or TCP, neither of which is well-suited to their needs. UDP contributes minimal latency, making it the recommended transport to meet the strict latency bounds of real-time applications [10], but provides limited support to applications, and is commonly blocked by enterprise firewalls. TCP prefers reliability to timeliness, and its congestion control tends to drive up queueing delay, but is often the only transport that can pass through middleboxes on the path. Accordingly, and despite its many problems, TCP is rapidly becoming the de facto transport for multimedia traffic.

In this paper, we engineer TCP Hollywood in response to these trends. TCP Hollywood is an unordered and time-lined transport protocol, that is wire compatible with standard TCP, but eliminates two sources of transport-induced latency, and provides reliability semantics that better suit real-time multimedia applications. Specifically, TCP Hollywood: 1) removes head-of-line blocking at the receiver and delivers received data to the application immediately, irrespective of ordering; and 2) relaxes reliability to respect time lines provided by the application, so only data that will arrive in time is retransmitted, otherwise retransmissions carry new data. The combination of both design elements reduces latency and introduces message-oriented semantics, allowing TCP Hollywood to express inter-dependencies between messages. Crucially, TCP Hollywood is wire-compatible with TCP, and incrementally deployable on the public Internet.

Our implementation consists of an intermediate logic layer that sits between the application and the kernel. Extensions in the TCP stack facilitate out-of-order delivery, and can be enabled or disabled via socket options. Messages are delineated in the logic layer using timing and dependency information from the application, and COBS-encoded [11] to survive re-segmentation that may occur in the network. We introduce the concept of inconsistent retransmissions: if the round-trip time (RTT) estimator indicates that a message will arrive too late to be useful, or if a message depends on a previous unsuccessfully transmitted message, then TCP Hollywood can exploit re-transmission slots to send new data and avoid retransmitting useless data. The semantics of TCP are maintained by preserving the sequence numbers in retransmitted segments, whether inconsistent or not. We develop an analytical framework to model the value of a retransmission against the buffering and processing time of data at the receiver-side. Our analysis reveals a wide range of RTT values where standard TCP retransmissions will arrive too late to be useful. We use this model to validate TCP Hollywood, and show that it handles retransmissions correctly.

ISBN 978-3-901882-83-8 © 2016 IFIP

Our contributions are as follows. After reviewing the rationale and requirements in Section II, we design a TCP-compatible architecture and application programming interface (API) that eliminates transport related, but not congestion control related, delay from TCP in Section III (this can be used with any of the existing proposals to reduce congestion control related delay, such as active queue management [12], [13] or delay-based congestion control [14], [15]). We develop an analytic framework in Section IV to determine the value and content of retransmitted data. We outline our implementation in Section V, alongside experiments to demonstrate ease of deployment. Related work and concluding remarks are provided in Sections VI and VII, respectively.

II. Rationale and Requirements

We begin by considering in more detail the requirements and rationale for an unordered and time-lined transport protocol. It is instructive to establish TCP as the foundation from which to build, and understand its negative impact in the context of live and interactive media.

Our primary design goal is to improve performance over TCP *for real-time traffic*, while maintaining deployability on the scale of TCP and UDP. Ossification of the transport layer means this goal can only be achieved by using TCP or UDP as a substrate. This is a limitation that exists in the Internet because of middleboxes that process packets based on static views of what is a valid transport. The operation of these middleboxes places two constraints on transport protocols [16]. First, only payloads marked as TCP or UDP are marked as valid; payloads carried by other transport protocols are often rejected. Second, middleboxes may reject valid TCP packets that don't conform to some limited subset of the TCP protocol that is understood by the middlebox [17]. For example, packets with the SACK (selective acknowledgement) extension might be rejected by a middlebox that doesn't understand that extension, and expects only regular ACK packets. In this restricted domain, reliability and congestion control are desirable features, that are difficult to implement at the application layer, and have forced TCP to emerge as the protocol of choice for real-time multimedia, despite struggling to meet latency bounds.

Our secondary goal is to minimize the transport-induced latency on applications. With TCP selected as the substrate, it remains to determine the appropriate modifications to meet our latency goals. TCP introduces latency in part because of the nature of its congestion control dynamics, and in part by providing an ordered, reliable, delivery model using head-of-line blocking and retransmissions. The former can be addressed using active queue management and/or delay-based congestion control algorithms, and has been widely studied. The latter issue is more applicable for real-time traffic, and is the subject of our work. Figure 1 shows the impact of head-of-line blocking: The loss of the third segment causes subsequent segments to be buffered at the receiver while waiting for the retransmission. Only when the delayed segment arrives can TCP deliver the in-order sequence to the application. The impact of retransmissions are exacerbated when they push segments outside of the window in which they are useful to the application. Effectively such segments are lost to the application, despite having been delivered to the host on time. It is these *late losses* that TCP Hollywood seeks to minimize.

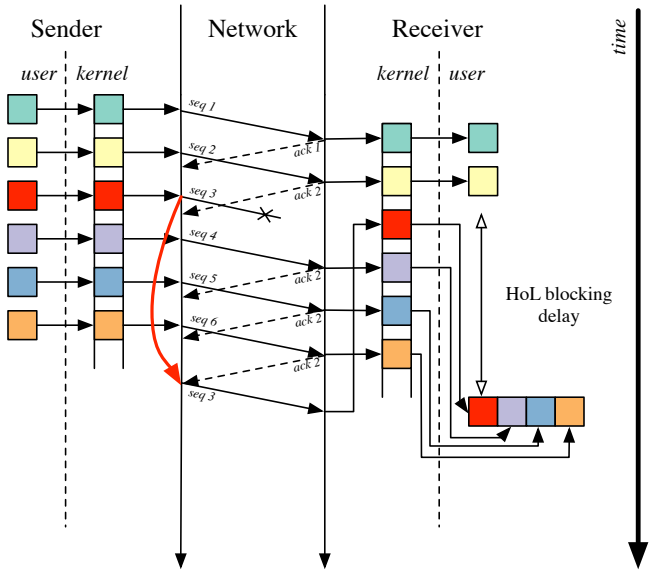


Figure 1. The interaction between head-of-line blocking and loss in TCP: multiple segments are delayed by a loss, and potentially delivered too late to be useful to the receiver

Two requirements follow. First, segments must be delivered as they arrive to eliminate head-of-line blocking. Second, retransmissions should be evaluated against timing information to ensure the delivery of useful data, by allowing *inconsistent retransmissions* to send new data in a segment that is retransmitted. So that applications can benefit from out-of-order delivery, a message-oriented abstraction is needed. Specifically, messages should be independently useful to the receiver [6]. With both a message-oriented abstraction and timing information, the collection of message dependency information follows. This increases the application-awareness of the transport layer.

III. Architecture and Design

TCP Hollywood has been designed to be deployable on the ossified Internet as it exists today, and to support partial deployments where only the sender or the receiver has been upgraded to support the TCP Hollywood extensions. The nature of the extensions we propose supports the former, while the latter is achieved by splitting the functionality between a user-space intermediary 'shim' layer and a set of extensions to the kernel TCP stack. The intermediary layer operates over either unmodified TCP, or with the TCP Hollywood kernel extensions enabled. The user and kernel components are represented in the overall architecture represented in Figure 2. In discussing the architecture it is useful to consider the sender separately from the receiver, and for each to consider the user-space intermediary layer separately from the kernel TCP extensions.

A. TCP Hollywood Sender architecture

The architecture of a TCP Hollywood sender is shown in the left hand side of Figure 2. The sender inherits the requirements identified in Section II to support a timed, message-oriented, transport abstraction, with inconsistent retransmissions.

The intermediary layer provides the message-oriented abstraction. It accepts a sequence of messages (i.e., datagrams,

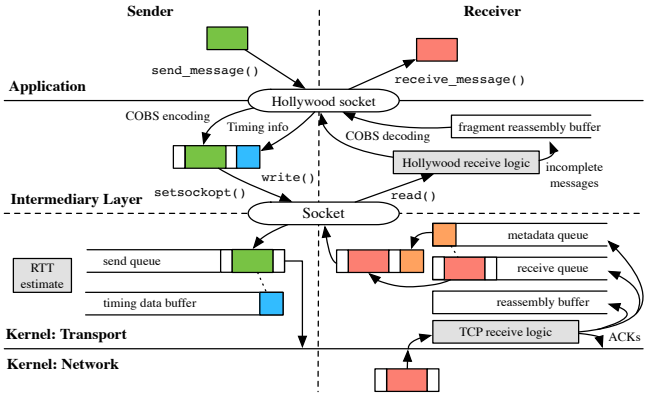


Figure 2. TCP Hollywood sender and receiver architecture

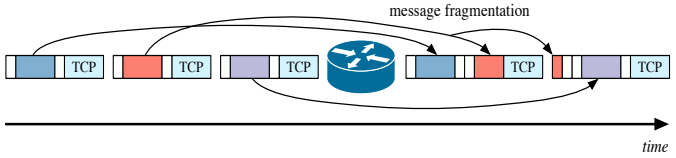


Figure 3. Encoding and framing with leading and trailing markers protects against middlebox re-segmentation; received segments can be properly decoded

rather than a byte stream) from the application, with optional timeliness and dependency information, to be delivered to the destination. The intermediary layer supports a sub-stream abstraction, allowing messages from multiple flows to be multiplexed on a single transport-level connection (similar to how multiple streams can be sent within a single SCTP association [7]). This can be used to cleanly multiplex audio and video flows onto a single connection, or to distinguish multiple layers of a stream encoded using scalable video coding [18], for example using H.264/SVC. The intermediary layer appends a sub-stream identifier to messages before they are encoded, framed, and passed to the kernel TCP sender, with a default sub-stream being reserved for flows where no sub-stream is specified. The application can provide timing or dependency data via the intermediary layer API. This is passed to the kernel alongside the encoded message, and used to determine whether inconsistent retransmissions are appropriate.

To support a message-oriented abstraction over a TCP byte stream, the TCP Hollywood flows must be resilient to re-segmentation or segment coalescing by middleboxes. Message integrity must be protected: messages received must have been sent, and only complete messages must be delivered. This is ensured by the intermediary layer, which frames messages with a leading and trailing marker. The effect is shown in Figure 3, where markers can be used to delineate messages irrespective of the segmentation. The intermediary layer encodes messages with consistent overhead byte stuffing (COBS) [19]; this efficiently encodes the stream to escape all zero bytes, allowing their use as framing markers, while still providing a transparent channel that can carry any message.

The TCP sender implementation in the kernel is modified to perform consistent segmentation, and to manage inconsistent retransmission by tracking message timing, deadline expiration, and dependencies. Consistent segmentation ensures that a single `write()` call made by the intermediary layer will generate a single TCP segment, provided the size of the segment does not exceed the MTU. This ensures each message is sent in a separate TCP segment, allowing the receiver to process it independently of other messages, reducing latency. This implies disabling Nagle's algorithm (i.e., setting the `TCP_NODELAY` socket option) to avoid unnecessary buffering – Nagle's algorithm would not provide a significant benefit to our target applications, where messages are large compared to their headers.

TCP retransmissions ensure reliability, but also inject latency that may cause late losses. A TCP Hollywood sender has the notion of *message expiry*: a message expires when (i) RTT estimates indicate the retransmitted message will arrive too late, or (ii) if the message depends on a previous message that was unsuccessfully delivered. Under these circumstances TCP Hollywood can send a new message using the same TCP sequence number space as a previously sent message, re-writing the remaining bytes in the TCP send buffer with new content. To support such *inconsistent retransmissions*, the intermediary layer passes messages down to the modified kernel TCP stack along with metadata to describe their deadline, dependency, and sub-stream. This is enabled by calls to the Berkeley Sockets API `setsockopt()` function. The metadata, with the exception of the sub-stream identifier, is never transmitted on the wire, but is held locally for each message for as long as the message is buffered (i.e., until all `ACK`s associated with a message are received). Our kernel extensions implement a separate buffer to hold per-message metadata.

The inconsistent retransmission logic is triggered when the standard TCP retransmission logic would be triggered by a triple duplicate ACK or timeout. Metadata for unacknowledged messages is then evaluated against the current RTT estimate, to determine whether the original message is to be retransmitted, or if an inconsistent retransmission is to be sent, replacing the original data with new content while keeping the same TCP sequence number. Since messages are framed and self-describing, a receiver can decode the inconsistent retransmission.

The latency benefits of inconsistent retransmissions will be quantified in Section IV. In the interim, we emphasize the message abstraction in this context: TCP Hollywood sends messages rather than bytes in a data stream. Consequently, a message may be composed of multiple fragments, split across TCP segments. To preserve the semantics at the receiver, fragments necessary to finish a partially received message are always retransmitted, but if no part of a message was received, it may be replaced with a new message when its containing TCP segment is retransmitted.

The processing overhead of TCP Hollywood at the sender is comprised of COBS encoding at the intermediary layer, and the maintenance of metadata in the kernel. COBS encoding requires a copy of the message to be made, but this could be eliminated by performing the byte stuffing as the message is being generated, as part of the multimedia encoding. Beyond this copy, COBS is "computationally cheap" [11]. In the kernel, the sender maintains metadata for each message, while the message could still be sent. Further processing, such as estimating whether a message will arrive on time, uses data already maintained by the kernel.

B. TCP Hollywood Receiver Architecture

The receiver-side architecture of TCP Hollywood is shown on the right hand side of Figure 2. Like the sender, it is composed of a user-space intermediary layer, and TCP extensions in the kernel receive path. The receiver supports message-oriented delivery, and additionally eliminates head-of-line blocking. The use of inconsistent retransmissions is invisible to the receiver.

The kernel initially processes incoming segments as would TCP. It generates the appropriate ACKs (e.g., duplicate ACKs for out-of-order or lost segments), and places segments into the reassembly buffer as usual. The on-the-wire response to each received segment is *identical* to that of TCP: ACKs (and SACK blocks, or other extensions, if negotiated) are generated in exactly the same way as standard TCP, and the congestion response is unchanged.

Where a TCP Hollywood receiver differs from standard TCP is that all segments, including those received out-of-order, are delivered to the intermediary layer in the order they are received, with no head-of-line blocking or reordering. As each segment arrives, a metadata structure is created to store its TCP sequence number. This sequence number is then appended to the segment as it is read by the intermediary layer. Sequence numbers are used by the intermediary layer to delineate messages that are encoded across multiple segments. Making segments available to the intermediary layer as they arrive is the only change needed to the kernel TCP code at the receiver.

The intermediary layer scans incoming segments for complete messages, delineated by the COBS framing. If consistent segmentation was used, and segments were not fragmented or coalesced in the network, then messages will correspond to TCP segments. Otherwise, incomplete message fragments are buffered in the fragment reassembly buffer awaiting missing fragments. The relative ordering of the bytes in message fragments is established using the TCP sequence number tag associated with received segments. As shown in Figure 2, complete messages are decoded and queued for delivery to the application. The API between intermediary layer and application is message oriented, and includes a message sequence number. This simplifies receiver processing compared to the TCP stream API.

The COBS decoding process is similar to that of the receiver, incurring an additional copy at the intermediary layer. In the kernel, our proof-of-concept implementation maintains a metadata structure to store the TCP sequence number and length of each incoming segment – data that would be otherwise lost. For incoming segments that are out-of-order, or arrive while there are segments in the reassembly queue, we make an additional copy (versus the TCP implementation) of the segment's payload, storing this with the segment's metadata. While this simplifies the implementation, it is not a requirement of the design: optimisation of our implementation could eliminate this.

C. Partial Deployments and Legacy TCP Compatibility

The TCP Hollywood intermediary layer is a user-space library that can run over a standard TCP implementation, using the Berkeley Sockets API, or on a modified TCP stack using the extensions we have described. If both sender and receiver support the kernel TCP extensions, the full benefit described above is achieved. However, the TCP Hollywood intermediary layer can also be deployed as part of an application, irrespective of the state of deployment of the kernel TCP extensions.

If only the receiver supports the TCP Hollywood kernel extensions, with a standard TCP sender, then the intermediary layer and application will benefit from avoidance of head-of-line blocking, but not from the latency reduction of inconsistent retransmission. Message oriented delivery will be supported, since COBS framing is generated by the intermediary layer at the sender, but COBS decoding may be less efficient since messages boundaries will be less likely to be aligned with segment boundaries.

If only the sender supports the TCP Hollywood kernel extensions, it will generate inconsistent retransmissions, and perform consistent segmentation as described, since both are invisible to the TCP layer of the receiver (compatibility with middleboxes is discussed in Section V-B). This will improve latency, and increase efficiency of COBS decoding, at the receiver, irrespective of whether the receiver has the TCP Hollywood kernel extensions.

If neither sender or receiver support the TCP Hollywood kernel extensions, the intermediary layers can communicate over a standard TCP connection. In this case, the message oriented abstraction persists, and applications can communicate using a TCP Hollywood socket to exchange messages, rather than byte streams, in a congestion controlled and reliable manner, although with no latency benefit over standard TCP.

IV. Latency Reductions and Analysis

TCP Hollywood reduces transport latency through support of inconsistent retransmissions, and by eliminating receiver-side head-of-line blocking. To quantify the benefits of these two techniques to the application, we begin by modelling the one-way transport delay, T_{owd}, as:

$$T_{\text{owd}} = T_{\text{sender}} + T_{\text{playout}} + T_{\text{rtt}}/2 \quad (1)$$

where T_{sender} is the time taken for the sender to capture, encode, and transmit a frame of media data. T_{playout} is the sum of the de-jitter buffering delay, and the time taken to decode and render a frame to the application at the receiver. Finally, T_{rtt} is the network round-trip time. With no loss of generality we assume broadly symmetric network paths in this analysis.[1]

The inter-frame interval of the media, i.e., the duration of media in each frame, is denoted by T_{framing}. We know that $T_{\text{sender}} \geq T_{\text{framing}}$, since a frame cannot be sent before it has been captured. Similarly at the receiver, if the media is to be decoded and rendered without gaps, then $T_{\text{playout}} \geq T_{\text{framing}}$. The time needed to encode and decode media is generally negligible in comparison to the framing interval, making $T_{\text{sender}} \approx T_{\text{playout}}$ a reasonable approximation in the absence of jitter. At the receiver, however, while the media decoding and rendering

[1]This assumption does not hold in ADSL and cellular networks with asymmetric downstream and upstream links. In these cases, our model misapproximates the application's upper bound on delay, shifting the line marked "Application Deadline" in Figure 4. While further analysis is needed to quantify the impact of this, it is clear that it does not change the broad conclusion of our analysis: that TCP Hollywood increases the usable region of retransmissions.

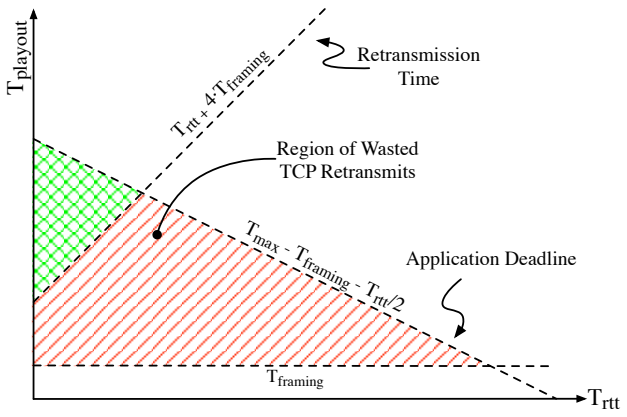


Figure 4. *Inconsistent Retransmissions* for real-time applications: TCP retransmissions may arrive too late to be used, if the play-out delay is set to meet the application deadline

time is generally small, the de-jitter buffer duration can be significant, and a similar approximation cannot be made.

The one-way transport delay contributes to an application's acceptable delay bound $T_{\max}$, such that $T_{\text{owd}} \leq T_{\max}$. For interactive applications, the delay bound is generally around 150ms [20], whereas streaming applications can accept longer delay bounds (around 0.5 seconds if channel surfing is to be supported; up to tens of seconds for on-demand streaming).

A. Utility of Inconsistent Retransmissions

TCP senders interpret a triple duplicate acknowledgement as an indication of packet loss, and retransmit the missing packet. It follows that the time needed by a sender to identify packet loss following a transmission has a lower bound of:

$$T_{\text{rexmit}} = T_{\text{rtt}} + 3 \times T_{\text{framing}} \tag{2}$$

At the receiver there is one additional framing interval to compensate for the interval that was lost with the original transmission. Assume media decoding and rendering take a negligible time. A retransmitted packet will arrive in time to be received and rendered to the application, provided:

$$T_{\text{playout}} \geq T_{\text{rexmit}} + T_{\text{framing}} \tag{3}$$

When $T_{\text{playout}} < T_{\text{rexmit}}$, retransmissions of the original packet will arrive after the data was scheduled to be rendered, and will be discarded by the application. This gives a lower bound on T_{playout} for standard TCP retransmission to be useful.

The corresponding upper bound is the maximum acceptable delay for the application,$T_{\max}$. If we assume media encoding delay is negligible, $T_{\text{sender}} \approx T_{\text{framing}}$. By combining these bounds, we see that standard TCP retransmissions will arrive in time to be rendered to the application, provided:

$$T_{\max} - T_{\text{framing}} - T_{\text{rtt}}/2 \geq T_{\text{playout}} \geq T_{\text{rtt}} + (3+1) \times T_{\text{framing}} \tag{4}$$

This inequality is shown graphically in Figure 4. The unshaded regions in Figure 4 fall outside of the feasible operating regime of the application and may be ignored, as they correspond to stalls in play-out or overall delay bound violations. The feasible operating regime is represented by the shaded regions that separate useful from wasteful retransmissions. The green cross-hatch highlights the region where standard TCP retransmissions arrive in time to be useful.

Wasteful TCP retransmissions are marked by the red-lined region in Figure 4. When the media play-out delay is less than the retransmission time ($T_{\text{playout}} < T_{\text{rexmit}}$) but satisfies the overall delay bound ($T_{\text{playout}} \leq T_{\max} - T_{\text{framing}} - T_{\text{rtt}}/2$), and is greater than the framing interval ($T_{\text{playout}} \geq T_{\text{framing}}$), then standard TCP retransmissions will arrive too late.This is where inconsistent retransmissions are useful: when a TCP retransmission will arrive too late to replace the original lost packet in this region. By contrast an inconsistent retransmission can use that retransmission slot to transmit the next unsent data segment. The lost packet is never recovered, but its sequence number is reused to send data that will be useful when it arrives.

B. Inconsistent Retransmissions and Real-Time Media

The benefits of TCP Hollywood can be quantified by substituting real-time traffic parameters into Equation 4. Consider interactive voice telephony. Widely deployed speech codecs typically use $T_{\text{framing}} = 20\text{ms}$ with a delay bound of $T_{\max} = 150\text{ms}$ [20]. Assuming media encoding delays are negligible, so that $T_{\text{sender}} = T_{\text{framing}}$, then the feasible region where standard TCP retransmissions arrive in time to be useful can be derived from Equation 4 as:

$$130\text{ms} - T_{\text{rtt}}/2 \geq T_{\text{playout}} \geq T_{\text{rtt}} + 80\text{ms} \tag{5}$$

which has valid solutions for T_{playout} provided $T_{\text{rtt}} \leq 33.33\text{ms}$. This round-trip time bound is low for wide-area networks. For example, TCP retransmission would be useful for calls from the authors' homes within Europe, but discarded during inter-continental calls. Figure 4 shows TCP Hollywood provides valid solutions for T_{playout} when $T_{\text{rtt}} \leq 220\text{ms}$, showing the utility of inconsistent retransmissions for this application.

For on-demand video streaming using the MPEG DASH framework, the framing interval and delay bounds are typically much larger. A typical deployment today might use an encoding segment size of $T_{\text{framing}} = 2\text{s}$, and an overall delay bound of $T_{\max} = 30\text{s}$. Assuming $T_{\text{sender}} = T_{\text{framing}}$, and substituting into Equation 4, this permits valid solutions for T_{playout} provided $T_{\text{rtt}} \leq 13.33\text{s}$, giving no benefit from inconsistent retransmission.

These two applications represent extremes in terms of latency bounds: voice telephony has tight latency bounds, while those of on-demand video streaming are relaxed. We analyse a third application: IPTV delivery using DASH. IPTV applications seek to minimise *zap time* (i.e., the total time taken between a viewer selecting a channel, and content from that channel being displayed). Bouzakaria et al. [21] show that end-to-end latencies – the time between encoding and decoding of a frame – of less than 240ms can be achieved using DASH. Using their techniques, segments are fragmented into 200ms chunks for delivery, giving $T_{\text{sender}} = T_{\text{framing}} = 200\text{ms}$. An overall delay bound of $T_{\max} = 1\text{s}$ allows for channel surfing to be supported. Substituting these values into Equation 4, we see that regular TCP retransmissions do not benefit this application for any RTT values. In contrast, inconsistent retransmissions in TCP Hollywood can be used when $T_{\text{rtt}} \leq 1\text{s}$.

Application	$T_{\max}$ (ms)	RTT Bound (ms)		Useful within a continent?		Useful intercontinental?	
		Standard	Hollywood	Standard	Hollywood	Standard	Hollywood
Voice telephony	150	33.3	220	Y	Y	N	Y
On-demand video	30000	13333.3	52000	Y	Y	Y	Y
Live video	1000	0.0	1200	N	Y	N	Y

Table I. SAMPLE TCP AND TCP HOLLYWOOD RTT BOUNDS REQUIRED TO MEET APPLICATION BOUNDS, HIGHLIGHTING INDICATES WHERE TCP HOLLYWOOD IS BENEFICIAL

Table I summarises the three applications considered. Utility of inconsistent retransmission is seen to depend on the latency bounds of the application. Interactive applications, where the overall latency requirements are tight, can strongly benefit from the ability to send new data in place of a retransmission, but those applications with relaxed latency bounds find less benefit.

C. Connecting Head-of-Line Blocking

If a packet is lost, then TCP will send a retransmission once a triple duplicate ACK is received. If standard TCP is used, then later segments will not be delivered to the application until the retransmission of the lost segment is received, potentially causing media play-out to stall. This is known as head-of-line blocking, as discussed in Section II.

The size of the play-out buffer relative to the round-trip time and media framing interval determines whether play-out stalls, or whether there is sufficient buffering to cover the retransmission delay. As was shown in Equation 3, if $T_{\text{playout}} \geq T_{\text{rexmit}} + T_{\text{framing}}$, then the retransmission will arrive in time to be played out, and no head-of-line blocking will occur.

However, if $T_{\text{playout}} < T_{\text{rexmit}} + T_{\text{framing}}$, then the retransmission will not arrive in time to be played-out. This will cause a 1-segment gap in the media play-out, since some data is missing (this occurs with both standard TCP, and with the TCP Hollywood extensions). If standard TCP is used, then the receiver may *also* suffer head-of-line blocking and be unable to access later segments, leading to a longer gap in play-out.

If the retransmission arrives less than one framing interval after it was scheduled to be played out, i.e., if $T_{\text{rexmit}} \leq T_{\text{playout}} < T_{\text{rexmit}} + T_{\text{framing}}$ then it will arrive before the following packet is to be played. In this case, there is no head-of-line blocking, and only a single packet gap occurs in play-out. If it is further delayed, such that $T_{\text{playout}} < T_{\text{rexmit}}$, then head-of-line blocking will cause one or more later frames to also miss their play-out.

A graphical representation is provided by Figure 5. The yellow cross-hatch region in Figure 5a is the region of T_{playout} values where blocked segments will be made wasteful. The details are labeled in Figure 5a by numbered events, with associated time-lines in Figure 5b. For a given value of T_{rtt} the process begins with a loss marked by the red '×'. The next frame arrives at ① and is held by TCP, as are all the segments that follow, awaiting the retransmission. For any size of T_{playout} at that moment ②, the retransmission will arrive too late. Upon arrival of the retransmission ③ TCP releases blocked segments to the play-out buffer. The duration of the head-of-line blocking that will be discarded by the play-out buffer is labelled as T_{HoL} in Figure 5, can be calculated as:

$$T_{\text{HoL}} = T_{\text{rexmit}} - T_{\text{playout}} = T_{\text{rtt}} + 3 \times T_{\text{framing}} - T_{\text{playout}} \quad (6)$$

The duration translates to N_{HoL} frames missing their play-out due to head of line blocking, and in addition to the retransmission that arrived too late, where:

$$N_{\text{HoL}} = \max\left(\left\lceil \frac{T_{\text{rtt}} + 3 \times T_{\text{framing}} - T_{\text{playout}}}{T_{\text{framing}}} \right\rceil, 0\right) \quad (7)$$

Finally, we remark on the grey shaded region in Figure 5a that occurs when that when retransmissions arrive past the acceptable deadline. From Equation 4, values of T_{playout} are upper-bound by the application deadline. Subsituting this into Equation 6 gives a lower bound on T_{HoL} of:

$$T_{\text{HoL}} \geq 3 \times T_{\text{rtt}}/2 + 4 \times T_{\text{framing}} - T_{\max} \quad (8)$$

As T_{rtt} increases under TCP, so too does T_{HoL}, and with it the fragility of the real-time connection. While a TCP retransmission under these circumstances will always arrive too late, the TCP Hollywood extensions eliminate T_{HoL}. In doing so the grey shaded region in Figure 5a, where real-time connections may be infeasible under TCP, are made viable with TCP Hollywood.

Our analysis identifies the value of inconsistent retransmissions, and the way in which they interact with head-of-line blocking. Specifically, it shows that removal of head-of-line blocking, via receiver side modifications to the kernel TCP stack, is necessary to make effective use of inconsistent retransmissions. For this reason, a full deployment of TCP Hollywood eliminates head-of-line blocking, to support latency reduction and improve good-put due to inconsistent retransmissions.

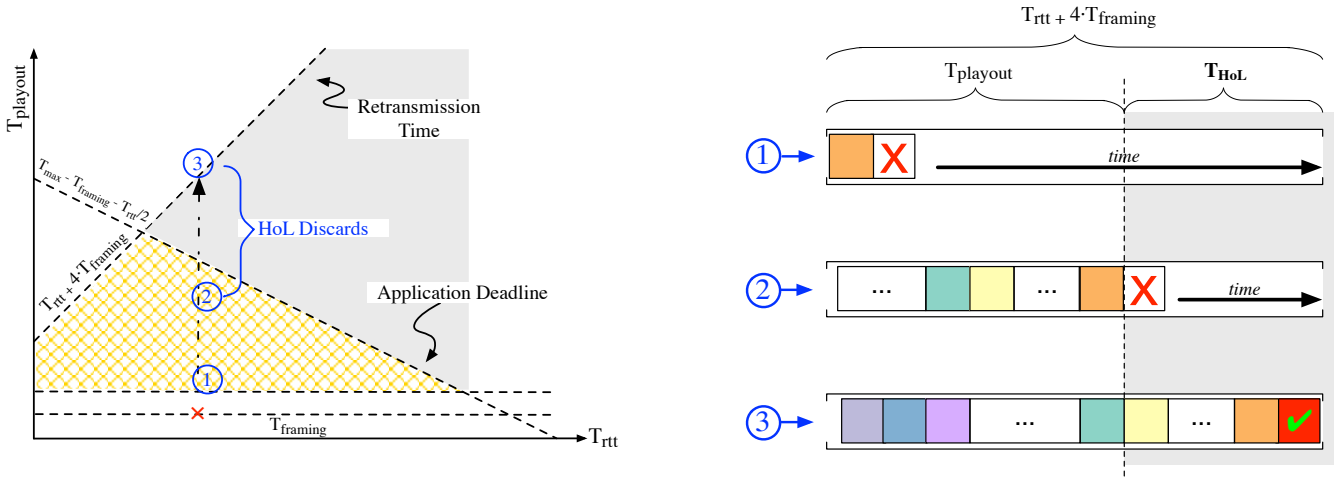
V. IMPLEMENTATION AND DEPLOYMENT

To evaluate our design, we have an implementation of TCP Hollywood that has been tested in fixed and mobile networks in the UK to evaluate ease of deployment.

A. Implementation

We implemented TCP Hollywood in the FreeBSD 10.1 operating system. The TCP modifications in the kernel impact approximately 300 lines of code, while the intermediary layer comprises 600 lines of user-space C code. The source code is available at http://dx.doi.org/10.5525/gla.researchdata.291.

The main implementation complexity in TCP Hollywood comes from the use of inconsistent retransmissions, since they cause the TCP RTT estimator to interact closely with the message deadlines and dependency tracking features of TCP Hollywood. Figure 6 shows sample results from a *dummynet* testbed used to validate our inconsistent retransmission implementation. These simulate a voice telephony scenario, like that



(a) $T_{playout}$ region where blocked segments will be delivered too late.

(b) Head of line blocking events between a loss and its retransmission.

Figure 5. *Head of line blocking* for real-time applications using regular TCP: for any given RTT and playout, segments that immediately follow a loss (1) are pushed past the acceptable deadline (2), and delivered as late as (3). The gap between RTT and playout is the duration of useful HoL blocked segments that become wasteful

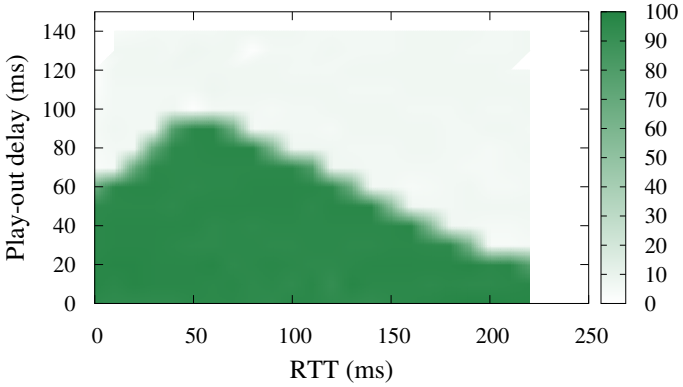


Figure 6. Implementation validation test. Shading shows the percentage of all retransmissions that are inconsistent at different combinations of play-out delay and RTT, for a VoIP scenario. This validates Figure 4

described Section IV-B, using 20ms framing, 120 byte payload per frame, and a maximum one-way delay of 150ms. The colours in Figure 6 show the fraction of retransmissions sent as inconsistent retransmissions, when subject to 5% random packet loss. RTT and playout delay were sampled at 10ms spacing across the entire range, with each point being repeated 5 times. We see that inconsistent retransmissions are triggered as expected, based on the analysis in Section IV. The step-like nature of the curve is due to the 10ms sampling interval. The fuzzy regions around the edge of the coloured space stem from limitations in the timer resolution that cause some degree of unpredictability in whether a retransmission will be inconsistent or not. This test shows that our implementation works as expected, but does not evaluate performance. A detailed evaluation of the implementation performance compared to the analytical results is for future work.

B. Feasibility of Deployment

We investigate the feasibility of deploying TCP Hollywood, using results from initial experiments with our FreeBSD implementation on residential and mobile networks in the UK.

TCP Hollywood ought to be entirely compatible with TCP. The only on-the-wire visible difference between a TCP Hollywood flow and a standard TCP flow appears within the payload data carried by inconsistent retransmissions. Recall from Section III-A that inconsistent retransmissions carry new payload data inside of segments with previously transmitted sequence numbers. This modification is invisible to receivers and middleboxes that only process TCP/IP headers, but is visible to middleboxes that use deep packet inspection if they compare the contents of a retransmitted packet with the original data. Depending on the configuration such behaviour may disrupt the connection. For example, a firewall may interpret inconsistent retransmissions as belonging to a man-on-the-side attack, and reset the connection.

We conducted experiments with a live deployment of TCP Hollywood to obtain an initial assessment on whether such middleboxes exist, and what impact they have. A TCP Hollywood server was setup on a public IP address, and configured to always send inconsistent retransmissions in lieu of the original data, so that all retransmissions contained new data with the same sequence numbers. The server was configured to listen on ports 80, 4001, and 5001. Port 80 is used by web traffic, and can be expected to be affected by middleboxes such as "transparent" caches and firewalls. We expect ports 4001 and 5001 to be less likely to be subject to interference by middleboxes, since they are not used by popular applications.

Clients were deployed across a number of access networks, operated by different service providers. Each client connected to the server, and received data. All incoming segments to the client host were recorded by `tcpdump`, then filtered by `iptables` to uniformly drop 5% of segments before reaching the TCP stack for traffic from ports 80 and 4001, leaving

traffic from port 5001 unaffected.[2] Each loss induced at the client triggered an inconsistent retransmission from the server. Remaining segments were passed up the stack to the client application, as normal. Data received by the client application was recorded, and compared against `tcpdump` logs from the server to identify the dropped segments, and to compare the payload data in the dropped segments with that sent in the original packet and in the inconsistent retransmission. This allows us to see what segments have been dropped, and to confirm that both the original and retransmission cross the path between client and server, and whether the inconsistent retransmission was delivered.

The evaluation was conducted using clients in 14 different locations in the UK, connecting to a server located at the University of Glasgow. The clients connected via eight different fixed-line residential ISPs (Andrews & Arnold, BT, Demon, EE, Eclipse, Sky, TalkTalk, and Virgin), and four mobile operators (EE, O2, Three, and Vodafone). All of the fixed-line residential ISPs successfully delivered the inconsistent retransmissions. In contrast only one out of the four mobile operators delivered inconsistent retransmissions. The three remaining mobile operators delivered the original segments instead, while the server saw no corresponding segment loss. The observed behaviour is consistent with a transparent split-connection TCP performance enhancing proxy cache that intercepts and responds to ACKs from the client on behalf of the server. On two of the three providers, this caching behaviour was seen on both port 80 and port 4001, while the other provider appeared to operate a cache on port 80 only.

Crucially, TCP Hollywood continued to operate whether or not the provider middlebox was present in the network. At no time did connections suffer a reset, and the use of the TCP Hollywood extensions did not affect connectivity or performance. Middlebox manipulations such as caching are designed to be transparent, leaving the client to believe it is interacting with a standard TCP server. Recall from Section III-C that TCP Hollywood is designed for partial deployment. This experiment provides evidence that TCP Hollywood continues to deliver messages and eliminate head-of-line blocking, even when inconsistent retransmissions are absent. In the worst-case, performance is the same as TCP without our extensions.

The set of networks tested is by no means exhaustive. Further, and larger scale, evaluation is needed to build evidence that inconsistent retransmissions are deployable. Previous studies provide room for optimism, however. Honda et al. [17] investigated deployment of TCP modifications with regards to middlebox interaction, from 142 networks in 24 countries, in early 2011, including inconsistent retransmission measurements taken over a large number of paths, with path diversity. Their observations mirror ours: the majority of paths deliver inconsistent retransmissions as expected, while a small number deliver the original instead. They also observed connection resets on one path, representing less than 1% of paths evaluated.

[2]Given that our goal is to test the ability to deploy TCP Hollywood, rather than performance, we are only concerned with creating sufficient loss to trigger inconsistent retransmissions. A high *un-correlated* drop rate enables TCP to survive where it would fail against correlated drops. The ensuing reduction in throughput translates to reduced loss due to congestion. Thus the client is more likely to see both the original transmission and its retransmission.

VI. Related Work

The immediate precursors of TCP Hollywood are the Minion protocol suite [22] and TL-TCP [23]. The Minion protocol suite includes uTCP, which proves a COBS-encoded user-space datagram abstraction atop TCP, with prioritization and out-of-order delivery. uTCP also provides an API that enables applications to replace existing datagrams in the transmission buffer before they are sent. Datagrams that have already been sent (i.e., those being retransmitted) cannot be replaced. The authors acknowledge this as a conservative design choice, made to ensure middlebox interaction.

Our wire compatibility experiments from Section V-B, and those of Honda et al. [17], indicate that inconsistent retransmissions are possible, but that the integrity of the sequence space needs to be preserved. The need to consider middlebox interaction with new or modified protocols is underscored by the design, and success, of Multi-Path TCP [24]. The design of TCP Hollywood builds on a number of protocols, and tweaks to TCP, that are unlikely to be deployable.

TL-TCP marks the first appearance of time-lines and inconsistent retransmissions [23]. The underlying mechanism works by injecting gaps into the sequence space. This modification is observable by middleboxes, and so is unlikely to be deployable. TCP Hollywood builds on TL-TCP, and related protocols, but does so while focussing on deployability. As discussed in Section III, we minimise changes to the wire protocol to maximise compatibility with middleboxes.

Transport protocols that rely on application-layer metadata to improve performance include Partially Error Controlled Connection (PECC) [25] and PRTP-ECN [26]. Other protocols such as SCTP [7] and DCCP [8] were engineered to broaden the delivery models offered by the transport-layer. Despite standardization and deployment in mainstream operating systems, their use is hampered by a lack of middlebox support.

Liang and Cheriton in [27] note that loss can be more detrimental to streaming application performance than jitter. On-demand streaming applications, for example, can effectively hide jitter from the application but are unable to tolerate loss. The authors present a modified TCP, TCP-RTM, that allows receivers to read beyond a gap in the receive buffer. The sequence numbers in the gap are ACKed, preventing their retransmission by the sender. Applications read from the socket at a predetermined play-out rate offset by some delay. There are no changes to TCP itself; instead, the interaction between application and receiver buffer is modified. Selective negative ACKs (NACKs) allow senders to be informed of the segments that were skipped over.

Deadline-aware TCP is a modified TCP specifically for datacenters, and implements flows with soft time constraints [28]. The modifications allow for the TCP window size and congestion back-off to be varied based on the flow congestion deadline. Flows with imminent deadlines benefit from larger windows. As the network becomes congested, flows will tend to complete closer to their deadlines. The modifications require ECN support in the network, and a modified TCP sender. Requiring ECN support effectively prevents deployment outside of datacenters.

QUIC (Quick UDP Internet Connections) [29] is a transport-layer protocol implemented atop UDP. It incorporates a number

of latency-reducing techniques (e.g., large initial data transfers, low RTT setups) that are slowly migrating to TCP. Its use of UDP as a substrate provides an interesting contrast to our choice of TCP. The main motivation being the ability to deploy without kernel modifications, and so QUIC is implemented entirely in userspace. The flexibility of a userspace implementation comes at the cost of universal deployment, since the initial estimates by the QUIC authors show around 5-10% are behind UDP-blocking firewalls. Upon detection of a blocking device QUIC is forced to fall back to TCP. The analysis presented in Section IV shows that falling back to TCP Hollywood is better for latency-sensitive applications (such as those using QUIC), in certain network conditions.

The trade-off between a UDP-based protocol with fall-back to standard TCP, as chosen by the QUIC authors, and a slightly modified TCP variant, as we have chosen, hinges on ease of implementation and deployment. We believe our implementation is simpler, since we build on the TCP infrastructure, but acknowledge that this gives us less flexibility to evolve the protocol. Equally, we believe our implementation is likely to be more deployable, as it builds on TCP. Broader measurement studies, for both TCP Hollywood and QUIC, are needed to evaluate this claim, however.

VII. Conclusions and Future Work

In this paper we presented TCP Hollywood, a modified TCP for real-time multimedia. The analysis shows that our modifications are beneficial to applications with tight latency bounds, such as voice telephony and live video delivery. Further, we've shown that by limiting the wire-visible modifications, we can maintain TCP's widespread ease of deployment.

Future work includes real-world performance evaluation. Measuring the performance improvements, in terms of the increase in usable bytes delivered to the application, that TCP Hollywood provides to the applications analysed in real networks is key to validating the analysis in Section IV. Beyond this, we are exploring enhancements to TCP Hollywood that may further improve performance. For example, dependency information is currently used to determine when *not* to send a message, but it may be a cause *to* send a message, even if that message may not arrive in time to be played out, to allow future messages to be processed. Broader enhancements, such as integration with SACK or MP-TCP, should also be studied.

TCP Hollywood exists within a transport-layer protocol design space that is constrained by ossification. We have TCP and UDP as substrates, with little room for modification. Substrate selection presents trade-offs: TCP gives a wider deployment story than UDP, but depending on the desired functionality, receiver-side kernel modifications can be needed. These trade-offs may shift over time, as the network responds to large deployments of substrate-based transports. For example, QUIC is seeing non-trivial deployment by being included within Google's web browser, and may result in fewer firewalls blocking UDP. This is a long-term concern, however, and in the near future we believe that protocols like TCP Hollywood offer important advantages relating to middlebox traversal, that will make them easy and valuable to deploy. Our initial results show TCP Hollywood is deployable on *all* major fixed and mobile operators in the UK, and offers compelling latency advantages.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," IETF, June 2002, RFC 3261.

[2] C. Jennings, T. Hardie, and M. Westerlund, "Real-Time Communications for the Web," *IEEE Communications*, vol. 51, no. 4, Apr. 2013.

[3] M. Cha, P. Rodriguez, J. Crowcroft, S. B. Moon, and X. Amatriain, "Watching television over an IP network," in *Proc. Internet Measurement Conference*. ACM, October 2008.

[4] T. Stockhammer, "Dynamic adaptive streaming over HTTP – standards and design principles," in *Proc. MMSys*. ACM, February 2011.

[5] Cisco, "Visual Networking Index: Forecast and Methodology, 2012-2017," White Paper, May 2013.

[6] D. D. Clark and D. L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," in *Proc. ACM SIGCOMM*, 1990.

[7] R. Stewart, "SCTP," RFC 4960, IETF, Sep. 2007.

[8] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, IETF, Mar. 2006.

[9] S. Hätönen *et al.*, "An Experimental Study of Home Gateway Characteristics," in *Proc. Internet Measurement Conference*. ACM, 2010.

[10] C. S. Perkins, M. Westerlund, and J. Ott, "WebRTC: Media transport and use of RTP," IETF, Nov. 2014, work in Progress.

[11] S. Cheshire and M. Baker, "Consistent Overhead Byte Stuffing," in *Proc. ACM SIGCOMM*, 1997.

[12] K. Nichols and V. Jacobson, "Controlling Queue Delay," *ACM Queue*, vol. 10, no. 5, May 2012.

[13] N. Khademi, R. Ros, and M. Welzl, "The New AQM Kids on the Block: An Experimental Evaluation of CoDel and PIE," in *Proc. Global Internet Symposium*. Toronto, ON, Canada: IEEE, April 2014.

[14] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. IEEE Infocom*, Mar. 2004.

[15] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *Proc. SIGCOMM Conference*. London, UK: ACM, August 1994.

[16] S. McQuistin and C. S. Perkins, "Reinterpreting the Transport Protocol Stack to Embrace Ossification," in *Proc. IAB Workshop on Stack Evolution in a Middlebox Internet*, Zürich, Switzerland, Jan. 2015.

[17] M. Honda *et al.*, "Is it still possible to extend TCP?" in *Proc. ACM IMC*, Berlin, Germany, Nov. 2011.

[18] J.-R. Ohm, "Advances in Scalable Video Coding," *Proc. IEEE*, vol. 93, no. 1, pp. 42–56, Jan 2005.

[19] S. Cheshire and M. Baker, "Consistent Overhead Byte Stuffing," in *Proc. ACM SIGCOMM*, 1997.

[20] ITU-T, "One-way transmission time," Rec. G.114, May 2003.

[21] N. Bouzakaria, C. Concolato, and J. L. Feuvre, "Overhead and performance of low latency live streaming using MPEG-DASH," in *Proc. 5th Intl. Conf. Information, Intelligence, Systems and Applications*. Crete, Greece: IEEE, 2014.

[22] M. F. Nowlan, N. Tiwari, J. Iyengar, S. O. Amin, and B. Ford, "Fitting Square Pegs Through Round Pipes: Unordered Delivery Wire-Compatible with TCP and TLS," in *Proc. USENIX NSDI*, San Jose, CA, Apr. 2012.

[23] B. Mukherjee and T. Brecht, "Time-lined TCP for the TCP-friendly delivery of streaming media," in *Proc. IEEE ICNP*, 2000.

[24] C. Raiciu *et al.*, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP," in *Proc. USENIX NSDI*, vol. 12, 2012.

[25] B. Dempsey, T. Strayer, and A. Weaver, "Adaptive Error Control for Multimedia Data Transfer," in *Proc. IWACA*, vol. 92, 1992.

[26] K.-J. Grinnemo and A. Brunstrom, "Evaluation of the QoS offered by PRTP-ECN - a TCP-compliant partially reliable transport protocol," in *Proc. IWQoS*, Karlsruhe, Germany, Jul. 2001.

[27] S. Liang and D. Cheriton, "TCP-RTM: Using RTP for Real Time Multimedia Applications," May 2002, submission to IEEE International Conference on Network Protocols (ICNP).

[28] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-Aware Datacenter TCP (D2TCP)," in *Proc. ACM SIGCOMM*, 2012.

[29] J. Iyengar and I. Swett, "QUIC: A UDP-based secure and reliable transport for HTTP/2," Work in progress, IETF, Jun. 2015.

BLEST: Blocking Estimation-based MPTCP Scheduler for Heterogeneous Networks

Simone Ferlin,*†, Özgü Alay*
*Simula Research Laboratory, Norway
{ferlin,ozgu}@simula.no

Olivier Mehani,† Roksana Boreli†
†National ICT Australia (NICTA), Sydney, Australia
{first.last}@nicta.com.au

***Abstract*—With the widespread availability of multi-homed devices, multipath transport protocols such as MPTCP are becoming increasingly relevant to support better use of multiple connectivity through capacity aggregation and seamless failover. However, capacity aggregation over heterogeneous paths, such as offered by cellular and Wi-Fi networks, is problematic. It causes packet reordering leading to head-of-line (HoL) blocking at the receiver, increased end-to-end delays and lower application goodput. MPTCP tackles this issue by penalising the use of longer paths, and increasing buffer sizes. This, however, results in suboptimal resource usage. In this paper, we first evaluate and compare the performance of default MPTCP and alternative state-of-the-art schedulers, all implemented in the Linux kernel, for a range of traffic patterns and network environments. This allows us to identify shortcomings of various approaches. We then propose a send-window *BL*ocking *EST*imation scheduler, *BLEST*, which aims to minimise HoL-blocking in heterogeneous networks, thereby increasing the potential for capacity aggregation by reducing the number of spurious retransmissions. The resulting scheduler allows an increase by 12% in application goodput with bulk traffic while reducing unnecessary retransmissions by 80% as compared to default MPTCP and other schedulers.**

***Index Terms*—MPTCP, multipath, transport protocol, packet scheduling, head-of-line blocking, receive window limitation, heterogeneous networks**

I. Introduction

Multipath transport protocols, and particularly Multipath TCP, allow to better use the network resources available to multi-homed devices such as mobile phones. Two main advantages are envisioned: capacity aggregation across multiple links, and the ability to maintain connection if one of the path fails. Capacity aggregation is however challenging with heterogeneous paths, such as offered by cellular and Wi-Fi, in particular because of delay heterogeneity [1]. This heterogeneity results in packet reordering, leading to head-of-line (HoL) blocking, increased out-of-order (OFO) buffer use at the receiver and, ultimately, reduced goodput.

MPTCP's default scheduler, *minRTT*, is based on Round-Trip Time (RTT). minRTT starts by filling the congestion window (CWND) of the subflow with the lowest RTT before advancing to other subflows with higher RTTs. When one of these subflows blocks the connection, *e.g.*, due to head-of-line blocking, MPTCP's default scheduler retransmits the segments blocking the connection on the lowest-delay path and penalise longer (*i.e.*, higher-delay) paths that caused the issue [2]. This has a long-term impact on the CWND of these subflows, which are limited in their growth [3], leading to sub-optimal capacity aggregation, as higher-delay paths are underused [4]. As a rule-of-thumb, it is also recommended to increase the receive buffer size to further limit HoL-blocking situations [5].

The need for multipath transport protocol schedulers is known, and a number of proposals have been made and evaluated in the past [6]. However, in the specific case of heterogeneous paths, more care is required to avoid the issues discussed above. Such schedulers have been proposed in [7]–[9], based on the concept of sending packets out of order so they reach the receiver in order. There exists, however, no comparison of these schedulers to the MPTCP default scheduler in a consistent environment.

In this paper, we first offer a comparative study of the proposed MPTCP schedulers [7]–[9], by experimentally evaluating our Linux implementation of these algorithms. We evaluate their behaviour for different traffic types (Web, Bulk, CBR). The performance of these schedulers is compared to MPTCP's default scheduler as well as plain single-path TCP, in terms of application goodput (for bulk traffic), end-to-end delays (CBR) and completion time (Web). Based on observations in these experiments, we identify how the studied mechanisms offer the best performance, and what they fail to properly account for. We also take insight from the observations of [10] that not all subflows should be used at all times and, while scheduling is needed to complement pure congestion control, path selection and send buffer management are also primordial. We then propose a novel *BL*ocking *EST*imation-based scheduler, *BLEST*, which takes a proactive stand towards minimising HoL-blocking. Rather than penalising the slow subflows, BLEST estimates whether a path will cause HoL-blocking and dynamically adapts scheduling to prevent blocking. Although BLEST is designed for heterogeneous paths, we show in our experiments that it works as well as MPTCP's minRTT scheduler in homogeneous scenarios.[1]

The remainder of this paper is organised as follows. We present the background to this work, and show motivating examples in the next section. We describe our evaluation setup in Section III. In Section IV, we discuss our implementation of different schedulers [7]–[9] and compare their performance side-by-side with MPTCP's default scheduler. Based

ISBN 978-3-901882-83-8

[1]BLEST's code is available at http://nicta.info/mptcp-blest.

on observations in these experiments, we propose a proactive minimum-delay scheduler that can predict the send-window blocking risk, and schedule accordingly in Section V, and evaluate its performance in Section VI, both in emulated and real multipath environments. We finally offer some concluding remarks in Section VII.

II. Background and Motivation

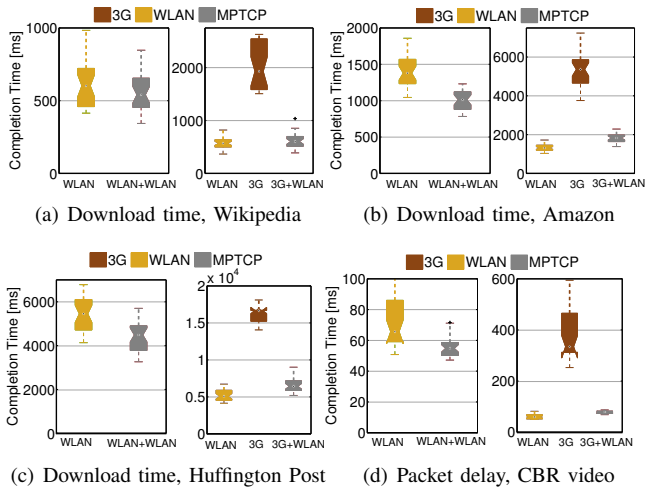
A. Multipath Transfer over Heterogeneous Paths

Multipath transport has be shown to provide benefits from bandwidth aggregation to increased robustness [2], [11]–[13]. Whenever the underlying network paths are homogeneous, MPTCP accomplishes its goals [14]. However, path heterogeneity can hinder achievement of MPTCP's goals, mostly due to the HoL-blocking which causes higher end-host memory usage and path bandwidth underutilisation [1], [3]. In MPTCP, the *scheduler* is the component that is responsible for the distribution of packets among the available paths. A well-designed scheduler that can dynamically adapt packet distribution based on the channel conditions to provide a better performance, both in terms of goodput and delay, is crucial.

MPTCP's default minRTT scheduler[2] first sends data on the subflow with the lowest RTT estimation, until it has filled its congestion window [2]. Data is sent on the subflow with the next higher RTT. In order to address the heterogeneity of the paths, a mechanism of *opportunistic retransmission and penalisation* (PR) has also been proposed in [2]. In order to quickly overcome HoL-blocking, opportunistic retransmission immediately reinjects segments causing HoL-blocking onto a subflow with an RTT lower than that of the blocking subflow which has space available in its congestion window. The penalisation mechanism also halves the congestion window of the blocking subflow to limit its use. [3] showed that MPTCP's PR does not behave well in some scenarios when path characteristics (*e.g.*, capacity, delay and loss rates) are significantly different. Penalisation of a long subflow (higher RTT) has a long-term detrimental impact on the performance: it will take longer for the subflow to increase its CWND, leading to underutilisation of the path and, ultimately, lower capacity aggregation.

In order to illustrate the challenges in heterogeneous scenarios, we ran experiments with constant bitrate (CBR) and web transfers, and contrast the results with homogeneous scenarios. In Figure 1, we observe that the amount of data and the path heterogeneity are the main factors determining the performance of MPTCP. MPTCP generally provides lower completion times, especially for websites with many objects. However, when the paths are heterogeneous in terms of delay and loss, as in the 3G+WLAN case, losses in the WLAN force MPTCP to use the 3G path, therefore MPTCP's completion time becomes higher than TCP on the WLAN path. Similarly, Figure 1(d) shows the same effect for the packet delay of a CBR flow: MPTCP's minRTT adequately leverages the aggregation of two homogeneous WLAN paths and reduces

[2]We base our work on MPTCP v0.90 throughout this paper.



(a) Download time, Wikipedia (b) Download time, Amazon

(c) Download time, Huffington Post (d) Packet delay, CBR video

Figure 1. Download times for selected websites, and application packet delay for CBR video traffic, both over MPTCP in **WLAN+WLAN** (left of each pair) and **3G+WLAN** (right of each pair) (CORE emulation, with background traffic, see III-1). MPTCP with heterogeneous paths (**3G+WLAN**) underperforms single-path TCP on the best (WLAN) path.

both delay and jitter; however, it doesn't perform as well as a single WLAN path when running over heterogeneous 3G+WLAN paths.

This goes against one of MPTCP's design goals: "[a] multipath flow should perform at least as well as a single path flow would on the best of the paths available to it" [5].

B. Schedulers for heterogeneous paths

Alternative multipath scheduling algorithms have been object of multiple studies [4], [10], [15]. In [6], the authors evaluated different scheduling strategies (pull, push and hybrid) focusing on implementation performance. They also considered how schedulers should cope with paths that have heterogeneous delay and/or capacities. They concluded that a scheduler must take both delay and capacity into consideration in order to effectively leverage multipath scenarios.

Later, [8] evaluated and extended the idea of a Delay-Aware Packet Scheduler (DAPS) [7] for MPTCP in order to overcome HoL-blocking due to path heterogeneity. In that work, the authors derived a rule-of-thumb for buffer size for MPTCP. [9] explored a more ambitious scheduler implementation, sending packets out of order so they arrive in order. They however included some simplications that expose vulnerabilities of the approach. For example, no consideration is given to segment reinjection if a certain path is blocking the connection.

These alternative algorithms were so far not extensively tested against MPTCP's default scheduler. The number of scenarios in which they were evaluated was also limited, and did not cover many scenarios (homogeneous vs. heterogeneous) and traffic classes. The differences in evaluation methods also make it difficult to accurately compare their performance. In the next sections, we address this issue by re-implementing these schedulers in the Linux kernel, and systematically evaluating their performance in a range of scenarios and traffic use-cases agains MPTCP's default scheduler.

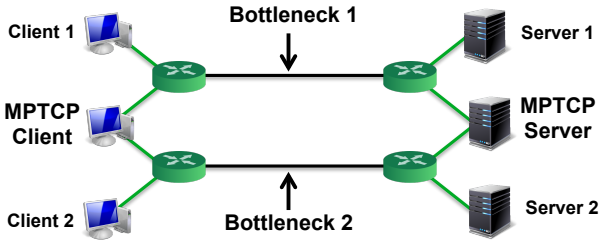


Figure 2. Emulation experiment setup

III. Measurement Setup

We used CORE [16] for the initial evaluation. CORE is a network emulator able to emulate a real network stack implementation within Linux containers, making it suitable to avoid simulation model simplifications. Figure 2 shows the emulation topology. Bottleneck 1 was loaded with background traffic from Server 1 to Client 1, and bottleneck 2 with traffic from Server 2 to Client 2. The link characteristics for WLAN and 3G links are set as follows.

- WLAN: Capacity=25 Mbit/s, Delay=25 ms, Loss=1%
- 3G: Capacity=5 Mbit/s, Delay=65 ms, Loss=0%

Based on measurements carried in real networks, the queue lengths at each router interface were set to 100 packets for WLAN and 3750 packets for 3G. The losses applied to the WLAN path are random.

1) Network and System characteristics: System settings are known to impact TCP's performance. In order to emulate realistic network scenarios, we used system settings close to the standard characteristics of each technologies. The TCP buffer sizes (send/receive) were set to be equivalent to widely known Android settings, that are configured as follows.

- Homogeneous (WLAN): 1024 KiB/2048 KiB.
- Heterogeneous (3G+WLAN): 1024 KiB/2048 KiB.

For bulk traffic experiments, we set both send and receive buffers to 16 MiB to evaluate MPTCP's aggregation capability.

To ensure independence between runs, the cached TCP values were flushed after every run. We focused on congestion avoidance; therefore, we discarded the initial phase for each experiment and analyzed a period of 90 s for bulk and constant bitrate (CBR) traffic. For single-path TCP flows, we used TCP-Reno, therefore, fairly compairing against MPTCP-OLIA.[3]

2) Application Traffic: We considered three different types.

a) Video Streaming: We considered constant bit-rate (CBR) video traffic with a frame size of 5 KiB on the application level and a rate of 1 Mbps. This is in line with the recent measurement studies [17] showing that more than 53% of the downstream traffic in North America is video streaming, and with other reports [18] predicting further increase,

b) Web Traffic: We selected three websites of different sizes, small, medium and large (see Table I), as a good set of typical website sizes. To mimic the behavior of a real browser downloads were performed with 6 concurrent connections.

c) Bulk Transfer: We completed the evaluation with the most common case for MPTCP — a buk transfer, of 64 MiB.

[3]TCP-Linux kernel 3.14.33 is used throughout our evaluations.

3) Background Traffic: A synthetic mix of TCP and UDP traffic was generated with D-ITG [19] as background traffic in order to create a realistic environment. The TCP traffic was composed of saturated sender and rate-limited TCP flows with a exponentially distributed mean rate of 157 pps. The UDP traffic was composed of UDP on/off flows with Pareto distributed on and exponentially distributed off times. Each flow has an exponentially distributed mean rate of 100 kbps in the heterogeneous scenario and 500 kbps in the homogeneous scenario. Packet sizes were varied with a mean of 1000 Bytes and RTTs between 20 and 100 ms. We repeated all experiment settings 50 times, in both emulation and real scenarios.

IV. Scheduling Against HoL-Blocking

In the following, we discuss both Delay-Aware Packet Scheduler (DAPS) [7], [8] and Out-of-order Transmission for In-order Arrival Scheduler (OTIAS) [9], evaluating them in common scenarios, and commenting on their implementation.

A. Delay-Aware Packet Scheduler (DAPS)

The DAPS algorithm was proposed in two versions. In [7], it pursues the goal to make segments arrive in order by planning which subflows the next segments should be sent over based on both the forward delay and CWND of each subflow. A schedule is created to span the least common multiple (LCM) of the forward delays $\mathrm{lcm}(D_i \in \{D_1, D_2, \ldots, D_n\})$. Algorithm 1 shows the main loop of the mechanism.

As an example, assume two subflows with similar capacities, but with a subflow having a forward delay ten times higher than the fast subflow. DAPS will derive the following schedule: segments 1...10 will be sent on the fast subflow, and segment 11 on the other subflow. Ideally, segment 11 will arrive right after segment 10, thereby avoiding HoL-blocking.

In [8], DAPS is formulated for a scenario with only two subflows (r_s and r_f). It is also a simplification of the original algorithm [7] as it does not take CWND asymmetry into account, only considering the subflows' RTT ratio (η) and the CWND of the fast subflow.

Since both algorithms are comparable, we consider only the original DAPS [7] in our evaluations. We ignore the simplifications presented in [8], as they were only introduced to ease the implementation in the ns-2 of CMT-SCTP.

B. Out-of-order Transmission for In-order Arrival Scheduler (OTIAS)

The OTIAS algorithm [9] is based on the idea of scheduling more segments on a subflow than what it can currently send. Queues may therefore build up at each subflow of the sender, under the assumption that these segments will be sent as soon

Table I
Web Traffic Generation

Domain name	Number of Objects	Size of Objects
http://www.wikipedia.org	15	72 KiB
http://www.amazon.com	54	1024 KiB
http://www.huffingtonpost.com	138	3994 KiB

Algorithm 1 DAPS [7]

$S_{max} \leftarrow 0$
for $P_i \in \{P_1, P_2, ..., P_n\}$ **do**
  $SEQ_{P_i} \leftarrow InitializeVector()$
end for
for $P_i \in \{O_1, O_2, ..., O_{\sum_i \in 1,2,...,n \frac{lcm(D_i)}{D_i}}\}$ **do**
  $SEQ_{P_i} \leftarrow Append(SEQ_{P_i}[S_{max}+1, S_{max}+C_i]$
end for
$t \leftarrow 0$
while $t < \text{lcm}(D_i \in \{D_1, D_2, ..., D_n\})$ **do**
  for $P_i \in \{P_1, P_2, ..., P_n\}$ **do**
    if $t \equiv 0 \ (mod \ D_i)$ **then**
      $Transmit(P_i, SEQ_{P_i}[\frac{t}{D_i}])$
      $S_{max} \leftarrow S_{max} + C_i$
    end if
  end for
  $t \leftarrow t + 1$
end while

Where:

- $\{P_1, P_2, ..., P_n\}$ set of paths
- $\{D_1, D_2, ..., D_n\}$ paths' respective forward delays
- SEQ_{P_i} seqnos of packets to be transmitted on P_i

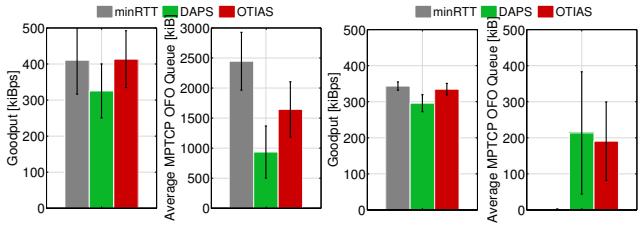
as there is space in the CWND for the subflow. When asked to schedule a new segment, the algorithm estimates its arrival time if sent over each subflow (T_i^j), and chooses the subflow with the earliest arrival time. The estimation is performed based on a subflow's RTT, its CWND, the number of in-flight packets and the number of already queued packets. If there is space in the CWND, the segment would be sent immediately, yielding an arrival time of approximately $RTT/2$ (assuming symmetric forward and backward delays). If the CWND is full, however, the segments will have to wait in the subflow's queue. Assuming a send rate of 1 CWND per RTT, the additional waiting time is calculated as $\texttt{RTT\_to\_wait}_i^j$. Algorithm 2 shows the main loop of the OTIAS mechanism.

Algorithm 2 OTIAS [9]

for `each available subflow` j **do**
  $\texttt{pkt\_can\_be\_sent}_j = cwnd_j - unacked_j$
  $\texttt{RTT\_to\_wait}_i^j = \left\lfloor \frac{\texttt{not\_yet\_sent}_j - \texttt{pkt\_can\_be\_sent}_j}{cwnd_j} \right\rfloor$
  $T_i^j = (\texttt{RTT\_to\_wait}_i^j + 0.5) \times srtt_j$
  if $T_i^j < min_T$ **then**
    $min_T = T_i^j$
    $\texttt{selected\_subflow} = j$
  end if
end for

C. Comparative evaluation of DAPS and OTIAS

Although DAPS and OTIAS have the same goal to reduce HoL-blocking, they follow different approaches: DAPS creates a schedule for the distribution of future segments into the available subflows for a *scheduling run* and follows this schedule until it is completed, after which planning for the next run is determined. On the other hand, OTIAS decides which subflow to use on a *per-packet* basis. It takes into account the RTTs and the queue sizes of the subflows at a given moment and it is closer to MPTCP's default scheduler (minRTT) in this respect, albeit taking into account more information from the subflows.



(a) **3G+WLAN** (b) **WLAN+WLAN**

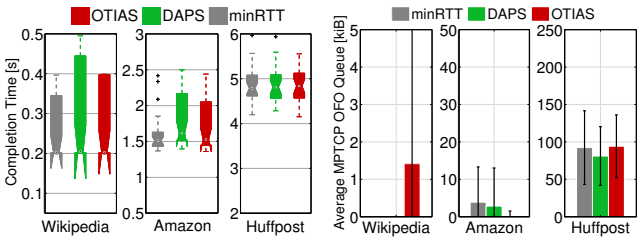
Figure 3. Goodput and OFO queue for bulk traffic between DAPS, OTIAS and minRTT.

OTIAS operates based on current data and is able to react more dynamically to network changes, where DAPS can only react to changes in the next scheduling run. OTIAS is however still less dynamic than MPTCP's minRTT since it builds up queues on the subflows. If a segment that had already been sent is blocking the connection, *e.g.*, it could be delayed or lost, the queued packets would linger at the sender more than assumed, disturbing the created schedule. Moreover, MPTCP's default scheduler retransmission mechanism, retransmitting a packet on the fastest subflow [4], is not applicable if a send queue exists for a subflow, as that segment would have to wait in the queue before retransmission.

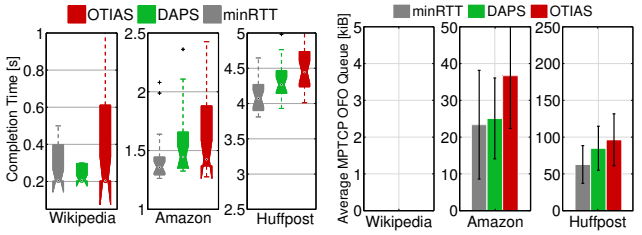
In the following we present an evaluation of DAPS and OTIAS against MPTCP's minRTT with bulk, web and CBR traffic through emulations. We look at application goodput for bulk transfers, completion times for web transfers, and average application delay for CBR traffic. In all cases, we also sample the maximum value of the out-of-order (OFO) queue every 10 ms during the experiments and present the results.

1) Bulk: Figure 3 shows DAPS, OTIAS and MPTCP's default scheduler goodput and OFO buffer size for bulk transfer in both 3G+WLAN and WLAN+WLAN scenarios. OTIAS provides a goodput increase of 6% but requires 35% less OFO buffer compared to MPTCP's minRTT. On the other hand, DAPS provides a goodput decrease of 27% and requires 65% less OFO buffer compared to MPTCP's default scheduler. In WLAN+WLAN scenarios, MPTCP's default scheduler has a 3.5% lower goodput compared to OTIAS, which on the contrary takes about 87% more OFO buffer. DAPS delivers goodput values of about 16% less compared to MPTCP's default scheduler with about 97% more OFO buffer.

2) Web: Figure 4 shows the completion times and OFO buffer sizes for DAPS, OTIAS and MPTCP's default schedulers in both 3G+WLAN and WLAN+WLAN scenarios. For 3G+WLAN, in Figure 4(a), all scheduler algorithms perform similarly in terms of completion time. However, for larger object sizes, we observe a larger OFO buffer size. In WLAN+WLAN, in Figure 4(c), DAPS and OTIAS struggle



(a) **3G+WLAN**, Completion time (b) **3G+WLAN**, OFO queue



(c) **WLAN+WLAN**, Completion time(d) **WLAN+WLAN**, OFO queue

Figure 4. Completion time and OFO queue for web traffic (Wikipedia, Amazon and Huffington Post) for DAPS, OTIAS and minRTT.

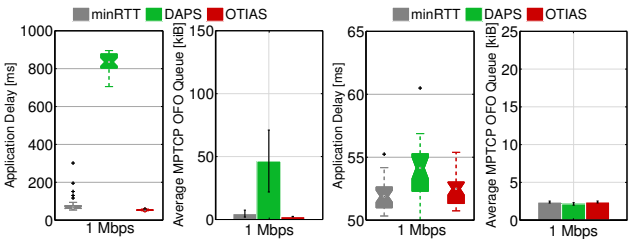
when both paths have higher loss rates, because DAPS cannot react quickly enough to changes on the paths, and OTIAS builds queues that also don't allow immediate reaction. While the losses on the WLAN paths cause higher OFO buffer size in WLAN+WLAN, the path heterogeneity is the main reason for the higher OFO size in 3G+WLAN.

3) CBR: Figure 5 shows the average application delay and the OFO buffer size for DAPS, OTIAS and MPTCP's default schedulers in both 3G+WLAN and WLAN+WLAN scenarios. Both 3G+WLAN and WLAN+WLAN yield higher application delay with DAPS. OTIAS can reduce the usage of the 3G subflow in the 3G+WLAN scenario, leading to improved application delay compared to MPTCP's default scheduler. However, for the WLAN+WLAN scenario, OTIAS provides higher delay values compared to MPTCP due to the lack of design for a reinjection mechanism. Moreover, MPTCP's default scheduler PR mechanism can partially overcome path heterogeneity in 3G+WLAN, where we can observe *burst* of packets on the 3G path, which lead to spikes in the OFO buffer, resulting in higher application delay.

D. Successes and Failures of Existing Algorithms

Overall, we observe that, although all state-of-the-art approaches address the challenges of multipath scheduling in heterogeneous scenarios, trying to overcome receive-window limitation and, consequently, HoL-blocking, they still fail in some typical use-case scenarios settings, *e.g.*, heterogeneous delays and/or loss rates, as well as with excessive delays due to buffering. Here, we comment on the strong and weak aspects of the state-of-the-art proposals just evaluated.

1) OTIAS: Although OTIAS can make decisions on a per-packet basis (subflow j and packet i) reacting fast and using current state from the network ($cwnd_j$ loop), it builds up queues on the subflows with lowest RTTs, regardless of their CWND state, *i.e.*, it does not restrict the scheduler if the



(a) **3G+WLAN** (b) **WLAN+WLAN**

Figure 5. Packet delay and OFO queue for CBR traffic for DAPS, OTIAS and minRTT.

CWND is full. In addition, the algorithm assumes symmetric forward delays ($OWD = RTT/2$), and scheduler reinjections (retransmissions) are not mentioned. While OTIAS can yield good results with heterogeneous RTTs, if the heterogeneity is too large and losses occur in one of the subflows, the algorithm will build up long queues in the subflows with lower RTTs, reducing their ability to overcome HoL-blocking. In homogeneous scenarios the OTIAS scheduler delivers lower performance due to not using both subflows as fully as MPTCP's default scheduler.

2) DAPS: The DAPS implementation is more complex, requiring more memory at run-time to keep the schedule run. Furthermore, DAPS is not able to react upon network changes in a timely manner due to long schedules arising from high heterogeneity in the subflow delays, *i.e.*, high LCM in Algorithm 1. Last but not least, DAPS will use all subflows that can send, even if a certain subflow's contribution is very low. This is the main contrast compared to both OTIAS and MPTCP's default schedulers, which can reduce the slow subflow contribution, if a faster subflow can sustain the required rate. This is particularly important for transfers where the sender is not saturated. Finally, similar to OTIAS, DAPS does not have a defined behaviour for scheduler reinjections.

V. BLEST: Blocking Estimation-based MPTCP Scheduler

Based on the observations from Section IV, we introduce a new algorithm, BLEST, addressing the challenges of reducing HoL-blocking, spurious retransmissions, and hence increasing application performance in heterogeneous scenarios. The scheduling is based on a new metric, estimating the amount of HoL-blocking, which might result from scheduling a packet on a give subflow.

For each new segment, MPTCP's default scheduler, minRTT, chooses the subflow with lowest RTT among all subflows ready to send, *i.e.*, with space in the CWND. When MPTCP detects that it cannot send new data due to a full send window (mirror of receive window at the sender), it will resend the segment blocking the fastest subflow, but only if it hasn't been sent on that subflow before. It will also penalise the slow subflow responsible for blocking, halving its CWND. The idea is to reduce its contribution preventing further HoL-blocking. Such an approach reduces the chance of HoL-blocking only for a limited amount of time. In other words,

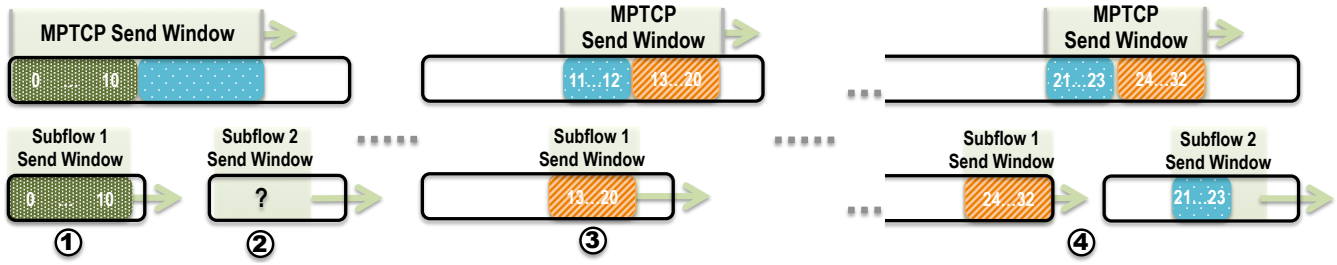


Figure 6. MPTCP example with BLEST: In ①, segments 0...10 are in flight on subflow 1, the subflow with lowest delay. In ② it is uncertain how many segments should be sent on subflow 2, which has a higher delay. While subflow 2's window could accommodate more data, only segments 11...12 are allocated, due to BLEST's blocking prediction. Here, minRTT would allocate as much data as fits into subflow 2's window given its CWND. In ③ subflow 1 can advance with segments 13...20, because 0...10 were acknowledged. At ④ both subflows can advance with MPTCP's send window with subflow 1 carrying segments 24...32 and subflow 2 carrying 21...23.

after the CWND was reduced by penalisation, the congestion control will start increasing it again, until a recurrence of blocking. Furthermore, the approach is reactive as it depends on blocking to trigger PR at the sender. The PR mechanism itself is detrimental in the long run, since it keeps the CWND of slow subflow artificially low.

To overcome the issues of the PR, we propose a proactive scheduler where we decide at packet scheduling time whether to send packets over the slow subflow or not. The decision is based on MPTCP's send window. MPTCP maintains a send window on its control-plane for each MPTCP connection, one level above the subflows. This window is necessary due to the full multiplexing among all subflows belonging to the same MPTCP connection. However, due to its scheduler design, if data is not acknowledged in one of the subflows, MPTCP's send window can be temporarily blocked, stalling the whole multipath connection.

BLEST assumes that a segment will occupy space in MPTCP's send window ($MPTCP_{\text{SW}}$) for at least RTT_S if it is sent now on subflow S, as illustrated in Figure 6. We assume that all segments in flight on S occupy space in the window for the same amount of time. This is a conservative assumption, as these segments can be acknowledged earlier. The remaining send window can be used by the faster subflow (*i.e.*, lower RTT subflow), F. This means that HoL-blocking would occur if F were not able to send due to lack of space in the send window because of S. Therefore, we estimate the amount of data X that will be sent on F during RTT_S, and check whether this fits into MPTCP's send window . To estimate X, we assume that for every RTT_F, its CWND grows by 1 (as it is done in congestion avoidance) and is always filled by the scheduler, as

$$rtts = RTT_S / RTT_F$$
$$X = MSS_F \cdot (CWND + (rtts - 1)/2) \cdot rtts$$

If $X \times \lambda > |M| - MSS_S \cdot (inflight_S + 1)$, the next segment will not be sent on S. Instead, the scheduler waits for the faster subflow to become available. Essentially, while minRTT always opts to use an available subflow, our scheduler is able to skip a subflow, waiting for a more advantageous subflow which can offer a lower risk of HoL-blocking, and the number of retransmissions that would have been consequently triggered.

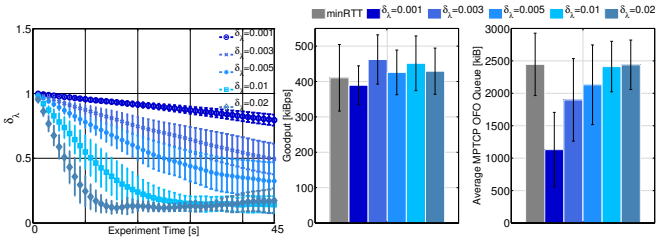


Figure 7. **3G+WLAN** and BLEST's λ parameter influence on bulk traffic with varying δ_λ=0.001, 0.003, 0.005, 0.01, and 0.02; compared against minRTT.

The estimate of X, however, can be inaccurate at times. To address this, we introduce a correction factor λ, to scale X. λ is adjusted as follows. HoL-blocking during one RTT_F is an event that triggers an increase of λ by δ_λ; the absence of HoL-blocking triggers a decrease by δ_λ. In the beginning of the connection we set λ=1.0, *i.e.*, no correction of the estimation.

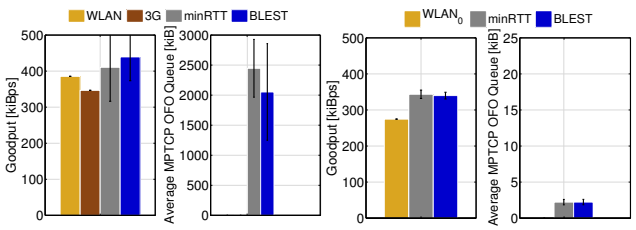
Figure 7 shows how λ changes over time in our scenario with different δ_λ. With $\delta_\lambda = 0.001$ we see that λ changes slowly towards a value that represents the reality on the (lossy WLAN) link. Note that X is over-estimated in the beginning of the transfer. Therefore, most of the traffic is sent over the WLAN link leading to a reduced goodput. However, in time, the estimate is corrected by λ to reach a steady value where the HoL-blocking is minimised.

On the left side, Figure 7 shows the first 45 seconds of a bulk transfer and how λ corrects the estimation (each dot in the plot curves show the average and standard deviation over 1s) of the rate of the faster subflow throughout the period. On the right side, Figure 7 shows the effect in the OFO buffer size and, consequently, in the goodput for different δ_λ. λ is corrected to lower values than its initial setting of 1.0, because the model does not incorporate losses.

VI. Evaluation

One of MPTCP's goals is to perform at least as well as TCP on the best path. For this reason, we compare MPTCP's default scheduler, minRTT, and BLEST against single path TCP on 3G and WLAN paths. We include both 3G+WLAN and WLAN+WLAN scenarios in our evaluation to illustrate the improvements in heterogeneous settings, while not impacting MPTCP in homogeneous scenarios. In the following we show emulations and real network experiments results.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



(a) 3G+WLAN (b) WLAN+WLAN

Figure 8. **3G+WLAN** and **WLAN+WLAN** scenarios for bulk traffic with minRTT, BLEST and TCP on 3G and WLAN.

A. Emulation Experiments

1) Bulk: Increasing application goodput for bulk transfer has been one of the most common ways to evaluate MPTCP's performance. Figures 8(a) and 8(b) compares the performance in terms of goodput and OFO queue size for minRTT and BLEST with bulk traffic in 3G+WLAN and WLAN+WLAN scenarios, respectively. In 3G+WLAN, we observe that BLEST reduces OFO buffer size by 19%, while it increases application goodput by 12%. Note that the MPTCP default scheduler's *penalisation and retransmission* (PR) mechanism has a particular negative impact in 3G+WLAN. As illustrated in Table II, MPTCP's PR mechanism can send up to 0.53 MiB retransmissions, to overcome blocking of the WLAN path. BLEST achieves better aggregation with less OFO buffer, saving up to 80% of retransmissions. In WLAN+WLAN, BLEST achieves similar application goodput with negligible OFO buffer size of 2.5 kiB compared to minRTT.

2) Web: The total download time is not a perfect metric as most browsers start rendering the page before the transmission is complete. However, we are focusing on the transport-level performance, and discard any browser-related optimisations. Figures 9(a) and 10(a) show the completion times for minRTT and BLEST for web traffic with different object sizes, see Table I. We also compare the OFO buffer size shown in Figures 9(b) and 10(b), and quantify the contribution of the additional subflow with smaller web objects, the amount of bytes transferred on each subflow relative to the transfer size, see Figures 9(c) and 10(c). In 3G+WLAN, for smaller web objects such as Wikipedia, the contribution of the additional subflow (3G) can be considered negligible, with only up to 2% of the total transfer. However, the small contribution of the 3G path for Amazon can cause an impact of up to 7% reduction in the completion time for BLEST compared to minRTT, see Table III and Figure 9(b). For Huffington Post, although the contribution of the additional subflow is still comparably low (about 2%), the completion time for BLEST is 6% lower than minRTT. In WLAN+WLAN, BLEST provides an improvement of 3% for Huffington Post and 2% for Amazon in completion times compared to minRTT. Overall, Table III illustrates the benefits of BLEST where the lowest completion time is achieved by the proposed BLEST algorithm for both heterogeneous and homogeneous scenarios for all the websites evaluated.

3) CBR: Live video has higher requirements of low latency compared to other forms of video streaming, *e.g.*, video on demand. Moreover, live video is more sensitive to network delay variations and, therefore, impacts the user experience the most. As we want to assess whether MPTCP could be used for applications other than bulk traffic, we evaluate live video performance that is more sensitive to latency. Figures 11(a) and 11(b) show the average application delay for minRTT and BLEST for CBR traffic with 1 Mbps. In the 3G+WLAN scenario, BLEST improved the application delay over minRTT by 8% for CBR (1 Mbps) and a slight improvement in OFO buffer size of 8% is also achieved, see also Table 11. In the same scenario and with the same application traffic, comparing BLEST to results shown in Figures 4, BLEST performed worse than OTIAS with CBR, because OTIAS completely discarded the 3G path. In contrast to that, DAPS keeps utilising the 3G path. In WLAN+WLAN shown in Figure 11(b), BLEST performed similar to MPTCP's default scheduler as expected.

B. Real Experiments

Finally, we validate the performance of the different schedulers with real-network experiments within the same topology as shown in Figure 2 for the emulation experiments, but now constructed over NorNet [20]. To generate background traffic, we use Virtual Machines (VM) from five commercial cloud service providers (2x in Europe, 1x in North America and 2x in Asia) connected via 100 Mbps links, as described in Section III, towards the server machine in Figure 12. We also use consumer hardware with a RaspberryPi connected to a home DSL provider via WLAN and another interface via 3G/3.5G to a mobile broadband operator. On the RaspberryPi side, background traffic from other connected devices congested both WLAN and 3G. The experimental setup is shown in Figure 12.

Table II
PENALISATION AND RETRANSMISSION MECHANISM TRIGGER IN 3G+WLAN WITH BULK TRAFFIC SHOWN IN FIGURE 3

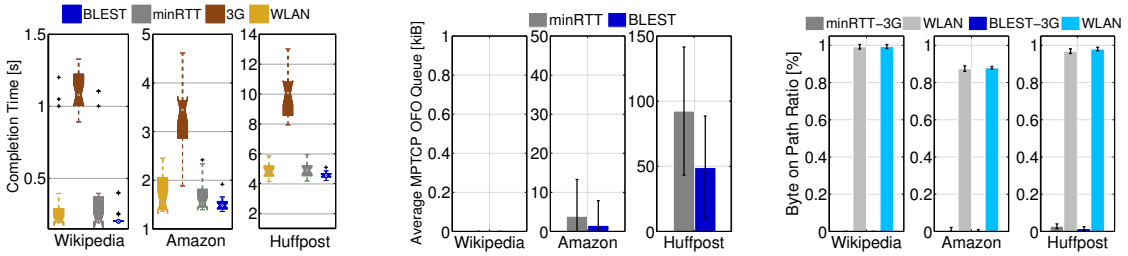
	Scheduler	Traffic	Retrans. Packets
3G+WLAN	**minRTT**	Bulk	366.37 **0.53 MiB**
	BLEST		70.3 **0.1 MiB**

Table III
AVERAGE WEB COMPLETION TIME, SEE FIGURES 4, 9, AND 10

			minRTT	OTIAS	DAPS	BLEST
Scenario	**Traffic**		**Completion Time [s]**			
3G+WLAN	Web	Wikipedia	0.421	0.392	0.435	**0.337**
		Amazon	1.60	1.724	1.789	**1.503**
		Huffington Post	4.87	4.858	4.932	**4.62**
WLAN+WLAN	Web	Wikipedia	0.398	0.4107	0.333	**0.324**
		Amazon	1.461	1.621	1.598	**1.456**
		Huffington Post	4.218	4.509	4.393	**4.114**

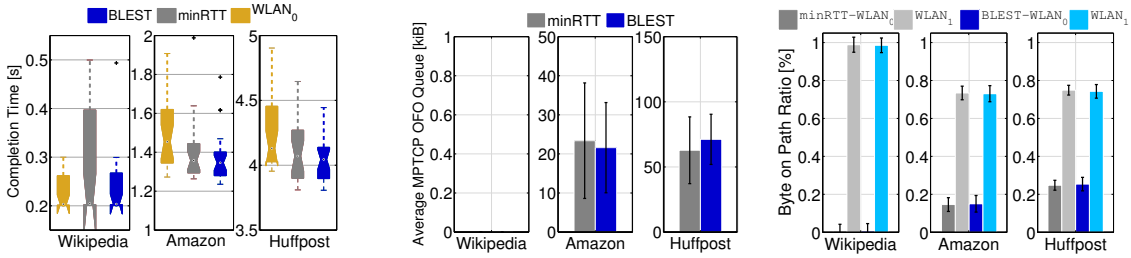
Table IV
AVERAGE CBR APPLICATION DELAY, SEE FIGURES 5AND 11

			minRTT	OTIAS	DAPS	BLEST
Scenario	**Traffic [Mbps]**		**Application Delay [ms]**			
3G+WLAN	CBR	1	68	53.2	843.7	**62.8**
WLAN+WLAN	CBR	1	52.18	53.49	54.02	**52.24**



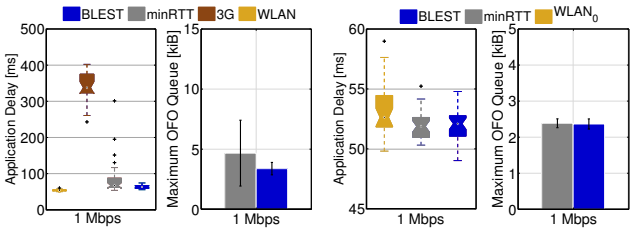
(a) Completion Time (b) MPTCP OFO Queue (c) Byte on Path: 3G and WLAN

Figure 9. **3G+WLAN** for web traffic with Wikipedia, Amazon and Huffington Post with minRTT, BLEST and TCP on 3G and WLAN.



(a) Completion Time (b) MPTCP OFO Queue (c) Byte on Path: $WLAN_0$ and $WLAN_1$

Figure 10. **WLAN+WLAN** for web traffic with Wikipedia, Amazon and Huffington Post with minRTT, BLEST and TCP on WLAN ($WLAN_0$ and $WLAN_1$).



(a) 3G+WLAN (b) WLAN+WLAN

Figure 11. **3G+WLAN** and **WLAN+WLAN** scenarios for 1 Mbps CBR traffic with minRTT, BLEST and TCP on 3G and WLAN.

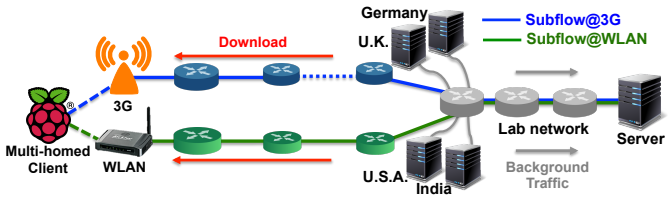
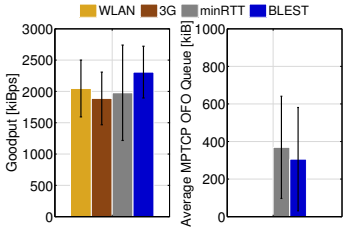


Figure 12. Real network experiment setup



(a) 3G+WLAN

Figure 13. **3G+WLAN** for bulk traffic in real experiments, see Figure 12.

In our experiments, we used the same parameters and settings from Section III as well as the same traffic from Section VI-A. We evaluate the performance of different schedulers under realistic network conditions, with real-network experiments, in a constructed non-shared bottleneck scenario as used in the emulation experiments shown in Figure 2.

1) Bulk: Figure 13 shows the application goodput and OFO buffer size for bulk traffic with minRTT and BLEST compared to TCP on 3G and WLAN paths. BLEST achieves on average 18% higher application goodput aggregation, while reducing the amount of retransmissions by more than 37%, see Table V, with a slight improvement in OFO buffer size of 3%.

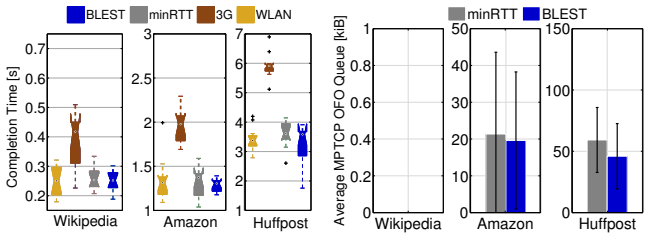
2) Web: Figures 14(a) and 14(b) show the completion times and OFO buffer sizes for the web transfers. With larger object sizes, BLEST reduces the completion time by up to 10%, while reducing the OFO size by up to 25%. Thus, MPTCP's performance with BLEST is closer to the WLAN path, only 3% worse than TCP on the best path (WLAN).

3) CBR: Figure 15(a) shows the the average application delay and OFO buffer size for the 1 Mbps CBR traffic with both minRTT and BLEST. BLEST improves the application delay by 11% while reducing the OFO size by more than 20%. We noticed, looking at single experiments, that MPTCP's default scheduler had small packet *bursts* sent over 3G, causing some spikes in the OFO buffer and, consequently, increasing the application delay. However, BLEST used the 3G path in the majority of the cases to send few single packets.

VII. Conclusion

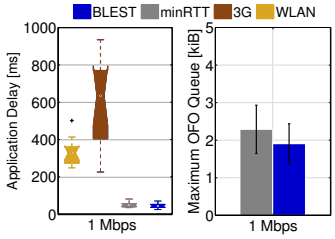
Path heterogeneity is rather the rule than the exception with MPTCP. Even subflows from a single machine can follow different paths to the destination with distinct delay, capacity,

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



(a) Completion time (b) MPTCP OFO queue

Figure 14. **3G+WLAN** for web traffic with Wikipedia, Amazon and Huffington Post in real experiments, see Figure 12.



(a) Delay and MPTCP OFO queue

Figure 15. **3G+WLAN** for CBR traffic in real experiments, see Figure 12.

and losses. Such path heterogeneity results in HoL-blocking at the receiver undermining MPTCP's overall performance. To overcome path heterogeneity, MPTCP follows a reactive approach and penalizes the subflows that cause HoL-blocking, through the *penalisation and retransmission* mechanism.

In this paper, we highlighted the limitations of such an approach for different application types in heterogeneous scenario through emulations and real-world experiments. Moreover, we have implemented and systematically evaluated scheduling algorithms aiming at mitigating this issue. We found, however, that neither was able to perform well in all multi-homing scenarios and traffic use-cases. We therefore proposed BLEST, a new scheduler based on a BLocking time ESTimation. Compared to previous proposals, BLEST directly considers the prospective HoL-blocking as a metric to minimise, and based on this metric, it dynamically selects whether it is worthwhile to schedule a packet on a specific subflow, or to ignore it. This allowed us to eliminate the *penalisation and retransmission* by implementing a more robust, proactive, scheduling metric. We evaluated our algorithm in emulated and real experiments with different application traffic (CBR, web and bulk). By comparing BLEST with minRTT, as well as the alternative DAPS and OTIAS, we showed that our approach outperfoms all algorithms across the presented scenarios, achieving its goal of reducing HoL-blocking, and consequently unnecessary retransmissions. This results in an increasing application goodput, lower packet delay and completion time, and reduced receiver buffer size.

Table V
PENALISATION AND RETRANSMISSION ALGORITHM RETRANSMISSIONS' OVERHEAD IN 3G+WLAN WITH BULK TRAFFIC SHOWN IN FIGURE 12

	Scheduler	Traffic	Retrans. Packets
3G+WLAN	**minRTT**	Bulk	33.42
	BLEST		21.3

For future work, we believe that both BLEST and OTIAS follow the right approach towards robust and effective scheduling for heterogeneous scenarios. We want to expand our evaluation with the method proposed in [5], add other elements of heterogeneity, *e.g.*, other network access technologies, evaluate different application performance metrics, *e.g.*, throughput aggregation versus delay constraints, increase the number of subflows and test the approach in mobility scenarios.

REFERENCES

[1] G. Sarwar, R. Boreli, E. Lochin, and A. Mifdaoui, "Performance evaluation of multipath transport protocol in asymmetric heterogeneous network environment," in *ISCIT 2012*.

[2] C. Raiciu, C. Paasch, S. Barré, A. Ford, M. Honda, F. Duchêne, O. Bonaventure, and M. Handley, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP," in *NSDI 2012*.

[3] S. Ferlin, T. Dreibholz, and O. Alay, "Multi-path transport over heterogeneous wireless networks: Does it really pay off?" in *GLOBECOM 2014*.

[4] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *ACM SIGCOMM Capacity Sharing Workshop (CSWS)*, 2014.

[5] C. Paasch, "Improving multipath TCP," Ph.D. dissertation, UCLouvain / ICTEAM / EPL, Nov. 2014.

[6] A. Singh, C. Goerg, A. Timm-Giel, M. Scharf, and T.-R. Banniza, "Performance comparison of scheduling algorithms for multipath transfer," in *GLOBECOM 2012*.

[7] G. Sarwar, R. Boreli, E. Lochin, A. Mifdaoui, and G. Smith, "Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer," in *WAINA 2013*.

[8] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: Intelligent delay-aware packet scheduling for multipath transport," in *ICC 2014*.

[9] F. Yang, Q. Wang, and P. Amer, "Out-of-order transmission for in-order arrival scheduling policy for multipath TCP," in *WAINA 2014*.

[10] B. Arzani, A. Gurney, S. Cheng, R. Guerin, and B. T. Loo, "Impact of path characteristics and scheduling policies on MPTCP performance," in *WAINA 2014*.

[11] S. Barré, C. Paasch, and O. Bonaventure, "Multipath TCP: From theory to practice," in *IFIP Networking 2011*.

[12] C. Paasch, G. Detal, F. Duchêne, C. Raiciu, and O. Bonaventure, "Exploring mobile/WiFi handover with multipath TCP," in *CellNet 2012*.

[13] C. Paasch, R. Khalili, and O. Bonaventure, "On the benefits of applying experimental design to improve multipath TCP," in *CoNEXT 2013*.

[14] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP," in *SIGCOMM 2011*, Toronto, ON, Canada.

[15] I. A. Halepoto, F. Lau, and Z. Niu, "Management of buffer space for the concurrent multipath transfer over dissimilar paths," in *DINWC 2015*.

[16] J. Ahrenholz, "Comparison of CORE network emulation platforms," in *MILCOM 2010*.

[17] Sandvine Intelligent Broadband Networks, "Global Internet Phenomena Report," Jul. 2013. [Online]. Available: https://web.archive.org/web/20141216103806/https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/sandvine-global-internet-phenomena-report-1h-2013.pdf

[18] Cisco Visual Networking Index, "Forecast and Methodology, 2014–2019," 2014. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[19] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[20] E. G. Gran, T. Dreibholz, and A. Kvalbein, "NorNet core — a multi-homed research testbed," *Computer Networks, Special Issue on Future Internet Testbeds*, vol. 61, pp. 75–87, Mar. 2014.

Multi-Flow Congestion Control with Network Assistance

Yannis Thomas, George Xylomenos, Christos Tsilopoulos and George C. Polyzos
Mobile Multimedia Laboratory & Department of Informatics
School of Information Sciences and Technology
Athens University of Economics and Business
Patision 76, Athens 10434, Greece
E-mail: {thomasi, xgeorge, tsilochr, polyzos}@aueb.gr

***Abstract*—A well-known technique for enhancing the performance and stability of content distribution is the use of multiple dissemination flows. Multipath TCP (MPTCP), the most popular multiflow protocol on the Internet, allows receivers to exploit multiple paths towards a single sender. Nevertheless, MPTCP cannot fully exploit the potential gains of multipath connectivity, as it must fairly share resources with (single-flow) TCP, without a clear understanding of whether the available paths do share any bottleneck links. In this paper, we introduce a hybrid congestion control algorithm for multisource and multipath transport that enables higher bandwidth utilization compared to MPTCP, while remaining friendly to TCP-like flows. Our solution employs (i) an in-network module that offers essential topological information and (ii) Normalized Multiflow Congestion Control (NMCC), a novel end-to-end congestion control algorithm. While NMCC is architecture-independent and the in-network module can be adapted for Multi-Protocol Label Switching (MPLS) or Software Defined Networks (SDNs), our prototype was implemented on the Publish-Subscribe Internetworking (PSI) architecture, which offers centralized path formation and source routing. Using an actual protocol implementation deployed on our test-bed, we provide experimental results which validate the effectiveness of our design in terms of performance, adaptation to shifting network conditions and friendliness to other flows.**

I. Introduction

Experience with content distribution indicates that multisource and multipath [1], i.e. the use of multiple sources and multiple paths to each source, respectively, can benefit both network operators and end users. First, the exploitation of multiple paths allows achieving higher throughput via bandwidth aggregation. Second, the use of multiple sources offers resilience to both link and source failures via path or source switching. As a result, multisource and multipath, collectively referred to as *multiflow* in this paper, provide load balancing and higher resource utilization, by spreading flows across more links and sources.

Multiflow transport is the focus of considerable research activity, due to the increase of multihomed devices, such as smartphones with WiFi, Bluetooth and Cellular connectivity. A significant body of research has focused on the side-effects of multipath, such as lack of TCP friendliness [2], [3], [4], [5], [6], [7]. This issue arises from the *uncoupled* congestion control scheme originally proposed for *Multipath TCP* (MPTCP), where an independent congestion window is used for each subflow. This causes the multiflow transfer of N flows to grasp up to N times more bandwidth than a single-path flow over the same bottleneck, thus causing the latter to starve. The current MPTCP congestion control algorithm achieves TCP-friendliness by limiting all subflows, so as to fairly share bandwidth with single-path flows. Nevertheless, blindly restricting multipath flows can lead to degraded resource utilization when friendliness is not an issue, for example, when a multipath flow exploits physically disjoint paths.

ISBN 978-3-901882-83-8

Efficient utilization of network resources is also the driving force of the *Publish Subscribe Internet* (PSI) architecture, an instantiation of the *Information-Centric Networking* (ICN) paradigm [8]. Following ICN principles, PSI bases communication on self-identified information items, rather than end-hosts. PSI also supports centralized path selection via a special network entity, the *Topology Manager*, and source routing via LIPSIN forwarding [9]. We have exploited these features in previous studies [10], [11], where we presented the *Multisource and Multipath Transfer Protocol* (mmTP), a multiflow transport protocol for PSI. The use of multiple paths to multiple sources was shown to greatly enhance the performance and resilience of mmTP over the, inherently unpredictable, PlanetLab testbed [11].

In this paper we focus on multiflow congestion control, proposing to exploit any available topological knowledge of the network to better balance performance and friendliness. Specifically, we present and evaluate a novel congestion control scheme for multiflow transport that consists of two independent modules: (i) *Normalized Multiflow Congestion Control* (NMCC), an end-to-end multiflow-aware algorithm, and (ii) an in-network mechanism to assist NMCC. NMCC is a simple, yet effective algorithm that manages bandwidth aggregation under the friendliness constraint, even in the face of heterogeneous paths and sudden changes in the congestion level. On the other hand, the in-network mechanism provides information about shared bottlenecks, thus allowing NMCC to adapt its behavior accordingly. Furthermore, we explain how our scheme can be adapted to IP networks operating over technologies utilizing centralized path computation components, including *Multi-Protocol Label Switching* (MPLS) and *Software Defined Networks* (SDNs).

The remainder of this paper is organized as follows. In Section II we summarize existing work on multiflow transport in IP and ICN networks. In Section III we briefly describe PSI and its features that allow us to realize selective friendliness. In Section IV we introduce our hybrid congestion control scheme, which consists of NMCC and the in-network assistance mechanism. In Section V we experimentally evaluate our design, using a prototype implementation. In Section VI we explain how the required in-network mechanisms can be provided by MPLS and SDNs. We provide our conclusions in Section VII.

II. Background Work

Multipath congestion control is an active research topic for both traditional IP networks and ICN clean-slate architectures. The common goal is maximizing resource utilization, in terms of exploiting the bandwidth available in multiple paths, while not harming competitive single-flow transfers, a constraint also known as *TCP-friendliness*.

A. TCP-friendliness

When a multiflow connection with N independent subflows competes against a single-flow connection for the same bottleneck link, the multiflow connection can be up to N times as aggressive as the single-flow one. While we usually say that the multiflow connection is not *TCP-friendly*, we will use the term *friendly* to imply *single-flow friendly*, defined as follows:

When a multiflow connection competes with a single-flow connection for the same network resource, the former should not acquire a larger share of that resource than the latter.

The price of friendliness is performance degradation: often, the bandwidth of the multiple subflows is not fully exploited, to prevent the starvation of single-flow connections. However, when the paths taken by each subflow are disjoint, meaning that we do not have multiple subflows sharing the same bottleneck link, this needlessly penalizes the multiflow connection.

B. Multiflow Congestion Control in IP

The *coupled*[1] congestion control algorithm of Multipath TCP (MPTCP) jointly tackles performance and friendliness [2]. MPTCP represents an evolution of TCP-Reno and EWTCP [5], adopting the slow-start and congestion avoidance phases per subflow, while also addressing multipath-specific problems, such as fair bottleneck sharing, *Round Trip Time* (RTT) mismatch and shifting network load. MPTCP manages its subflows under two constraints: (i) a multipath flow should achieve at least as much throughput as it would get with single-path TCP on the best of its paths and (ii) a multipath flow should grasp no more capacity on any path or collection of paths than a single-path TCP flow using the best of those paths. The second constraint, which assures MPTCP's friendliness towards unicast connections, compromises performance when friendliness in not an actual issue, for example, when the available paths do not share a bottleneck link.

Even though this decision may be far from optimal, it is imposed by the IP routing architecture. Due to the distributed, hop-by-hop routing of IP networks, a transport protocol cannot reliably detect whether the dissemination paths used are overlapping. As a result, its congestion control module cannot detect whether friendliness is an issue or not. There are some application-layer solutions for the end-to-end detection of shared bottlenecks [12], [13], but their efficiency is debatable. In [12] the authors detect shared bottlenecks based on the temporal correlation of fast-retransmit packets, while in [13] the authors evaluate both loss-based and delay-based correlation techniques, arguing that the loss-based technique is unreliable, while the delay-based methods require considerably more time for accurate results; also, the convergence time of the loss-based method is roughly 15 ms, which is unrealistically high for a general purpose multiflow protocol.

[1]We use the term *coupled* to refer to the final algorithm presented in [2].

C. Multiflow Congestion Control in ICN

In ICN networks, the location-based networking of IP is replaced with information-based routing and forwarding. These features can support more efficient transport patterns, such as multipath, multisource and multicast, since they pin transport paths on the physical topology. Thereupon, it is often proposed that ICN routers should assist topology-aware congestion control so as to better handle friendliness issues.

Along these lines, in [14] and [15] the authors discuss the design of transport protocols that pull data from multiple sources via multiple paths over the *Content Centric Networking* (CCN) architecture [8], exploiting congestion detection and control in the forwarding nodes. In [14] flow control and part of congestion control is managed by the receiver, but in-network congestion control is also present in the form of dynamic request forwarding: intermediate routers choose on-the-fly the most appropriate interface to forward each packet, shifting flows to less congested parts of the network. In contrast, in [15], traffic control is exclusively assigned to in-network nodes, which separate content (cache) from forwarding (queue) storage: each router maintains a per-flow queue with the *Deficit Round Robin* (DRR) scheduling policy to determine which packets must be dropped and/or connections must be rejected, based on link utilization and fairness constraints. The receiver uses a simple control loop, responding to explicit congestion signals from routers.

The stateful CCN-based approaches have some important disadvantages. First, CCN nodes face significant overheads: the estimation of link utilization for congestion detection in [14], [15] and the additional per packet state for fair queuing in [15] can impact their performance, making the achievement of wire speed forwarding doubtful. Second, distributed in-network congestion control has a delayed reaction to losses. While TCP rapidly detects lost packets via either out-of-sequence packets or time-outs, in [15] authors use explicit notifications to the receiver when a queue drops a packet; [14] introduces a novel time-out estimation function, which is not investigated with regard to its effects on the other CCN timers.

A different approach for enriching congestion control with topological information, involves an in-network notification system that can report the existence of shared bottlenecks.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

This notification system, which must be aware of both network structure and dissemination routes, should explicitly indicate path disjointness to the end-hosts, allowing them to apply friendliness mechanisms more selectively. This design offers accurate information without convergence delay and without stressing the core routers, which are the weaknesses of the IP and CCN solutions, respectively. PSI follows this approach, since routing takes place at a conceptually centralized in-network entity, the *Topology Manager*. We briefly discuss the PSI architecture in the following section.

III. Multiflow transport in the PSI architecture

A. The PSI architecture

In the PSI architecture, content objects are treated as publications, content sources as publishers and content consumers as subscribers. User programs exploit a publish/subscribe API for advertising and requesting information. A fundamental design tenet in PSI is the clear separation of its core functions [16]: (i) the Rendezvous function tracks available publications and resolves subscriptions to publishers, (ii) the Topology Management and Path Formation function monitors the network topology and forms forwarding paths and (iii) the Forwarding function handles packet forwarding [17].

Network nodes in a PSI network are classified into *Rendezvous Nodes* (RNs), *Topology Managers* (TMs) and *Forwarding Nodes* (FNs). The RNs receive and store the pub/sub requests and match publications with subscriptions of the same content. When matching takes place, the RN asks a TM to find the appropriate dissemination routes. The TM, which is aware of topology, network conditions and content characteristics, discovers the "best" path(s) and encodes them into LIPSIN identifiers [17]. LIPSIN forwarding, which is realized by the FNs, offers line-speed stateless source routing. Finally, the LIPSIN identifiers are delivered to the end-host applications that exploit them for direct communication, thus delegating congestion control to the network edges.

The centralized nature of the TMs raises concerns about PSI's feasibility, since they must compute paths for all network connections. However, recent work showed that a centralized intra-domain TM service is feasible: for a typical national-scale network provider in the UK, it was demonstrated that a reasonable number of TM instances with precomputed paths can efficiently cope with the resulting network load [18].

B. Multipath and multisource in PSI

We have presented a multiflow transport protocol for PSI in previous studies [10], [11], the *Multisource and Multipath Transfer Protocol* (mmTP). mmTP is a reliable protocol that supports multisource and multipath data transfers by exploiting PSI's source routing and centralized path selection. mmTP relies on a TM function that can discover multiple paths between a receiver and multiple senders. These paths are encoded in LIPSIN identifiers that are later sent to the end-hosts. Given that LIPSIN identifiers encode dissemination routes without unveiling the actual dissemination paths, or even the destination nodes, the end-hosts acquire a set of distinct options for requesting data, which may involve different publishers and/or different paths. Hence, mmTP provides a generic interface, transparently supporting any combination of multisource and/or multipath services.

The design of mmTP allows congestion control in two levels: (i) path selection by the TMs and (ii) path utilization by the end-hosts. Specifically, the TMs, which are aware of network conditions, select appropriate routes for load balancing and bandwidth aggregation. We have previously shown the gains of centralized path formation in [19], where we used QoS routing schemes to satisfy certain throughput and error rate constraints in PSI. Based on these routes, the end-hosts evaluate in real-time the performance of each path and adjust the amount of data to be delivered through it. The congestion control mechanism used at the end-hosts, which is derived from TCP, pushes complexity at the network edges, thus enhancing network stability and keeping forwarding stateless.

IV. Hybrid multi-flow congestion control

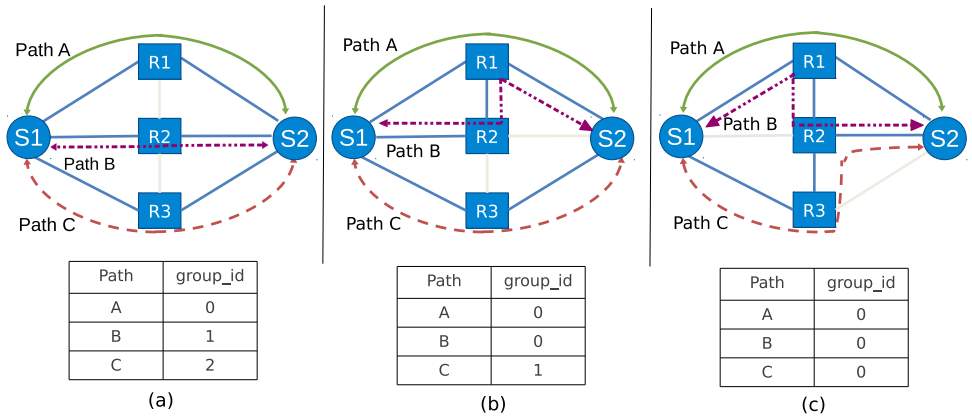
In this section we present a hybrid multiflow congestion control algorithm that enhances resource utilization without violating the friendliness requirement. Our novel congestion control scheme consists of two independent modules: (i) NMCC, an end-to-end multiflow-aware algorithm, and (ii) an in-network mechanism to assist congestion control. NMCC is simple, yet it outperforms the coupled congestion control of MPTCP in terms of friendliness in short transfers and performance in heterogeneous networks. The in-network mechanism exploits knowledge of shared bottlenecks to enhance the performance adaptation of NMCC.

A. Path Formation

The best case scenario for multiflow communication arises when all communication paths are physically disjoint, that is, they do not share *any* links or routers. In this case, each multiflow connection can use the same congestion control algorithm as single-flow connections. In contrast, when some subflows use paths which are not disjoint, their aggressiveness needs to be limited in order for them to remain friendly.

Path selection in PSI is performed by the TMs, whose operation extends beyond the scope of this paper. Our only requirement is that when the TMs return a set of paths encoded as LIPSIN identifiers, a $group\_id$ code should be added to each identifier so as to indicate non-disjoint paths. Specifically, all paths that share at least one link with each other (not necessarily *the same* link) are marked with the same $group\_id$. In general, for any given underlying routing mechanism, the in-network assistance mechanism must be able to signal to NMCC how the available paths are grouped by $group\_id$.

For example, Figure 1 shows three examples of path composition along with the corresponding $group\_id$ codes. In Figure 1(a) the three paths are disjoint, thus each path is marked with a distinct $group\_id$, whereas in Figure 1(b) paths A and B share a link, thus they have the same $group\_id$. In 1(c) Paths A and B share a link and paths B and C share a



Path	group_id
A	0
B	1
C	2

(a)

Path	group_id
A	0
B	0
C	1

(b)

Path	group_id
A	0
B	0
C	0

(c)

Fig. 1. Three different cases of path composition and their corresponding *group_id* codes: (a) Disjoint paths, (b) Paths A and B share one link, (c) Paths A and B share one link, paths B and C share another link.

different link; they still get the same *group_id*, to ensure that each path belongs to a single group. This simplifies operation, at the cost of losing some efficiency, since a congested link may only affect some of the paths in a group.

B. Window Management

When the available paths have different *group_ids* (i.e., they do not share any links), then window management does not consider friendliness: our algorithm creates a distinct TCP-like subflow for each path with an individual congestion window variable (*cwnd*), RTT-based loss detection mechanism, retransmission mechanism and *slow start* and *congestion avoidance* algorithms. Therefore, window management is similar to MPTCP's uncoupled congestion control scheme.

In contrast, when multiple paths have the have the same *group_id* (i.e., they share some links) our NMCC algorithm is used to maintain friendliness. NMCC differs from the coupled congestion control algorithm of MPTCP in two respects. First, coupled MPTCP only tries to limit its aggressiveness during congestion avoidance, while NMCC also considers slow start. Second, NMCC is simpler to operate than coupled MPTCP. The coupled MPTCP algorithm is a variant of TCP Reno, where aggressiveness is controlled by reducing the growth rate of the congestion window per RTT. This introduces complications when paths have different RTTS and shifting network loads, which are due to the use of a single-flow solution to a multiflow problem. On the other hand, NMCC exploits a well-known TCP-fairness issue, the fact that connections with higher RTTs are less aggressive [20], to ensure friendliness. Instead of reducing the growth of the congestion window per RTT, NMCC controls the congestion window by inflating the RTTs; this reduces complexity, simplifies friendliness during slow start and avoids multiflow-related issues due to RTT mismatch and sudden load and congestion shifts.

1) Congestion Avoidance: NMCC uses an inflated $RTT'_i \geq RTT_i$ for each subflow i to control window growth; the inflated RTT'_i makes the congestion window grow slower compared to a single-flow connection. We introduce a *friendliness factor* $m \geq 1$ so that $RTT'_i = m * RTT_i$, trying to approximate the two goals of fair bottleneck sharing: (i) the growth rate of all subflows sharing a link should be no more than that of a single-flow connection and (ii) the overall growth rate should not be less than that of the most aggressive single-flow connection. Since the most aggressive single-flow connection has the minimum $RTT_i = RTT_{min}$ and during congestion avoidance the growth rate of a single-flow connection is one packet per RTT, the rate increase intervals during congestion avoidance must satisfy the following equation:

$$\frac{1}{RTT_{min}} = \sum_{i=1}^{N} \frac{1}{RTT'_i} = \sum_{i=1}^{N} \frac{1}{m * RTT_i}$$

where N is the number of jointly controlled subflows. We can therefore estimate m using the following equation:

$$m = RTT_{min} * \sum_{i=1}^{N} \frac{1}{RTT_i}$$

To understand the friendliness factor m, consider a simple example. When the TM offers two paths marked with the same *group_id*, we initially set $m = 2$, the number of jointly controlled paths. Upon receipt of the first packet over each path, the RTT_i's are updated and m is re-calculated. If $RTT_1 = 50$ ms and $RTT_2 = 100$ ms, then $m = 1.5$, so $RTT'_1 = 75$ ms and $RTT'_2 = 150$ ms, therefore NMCC will increase its overall congestion window by three *maximum segment sizes* (MSS) during a period of 150 ms: two MSS from the first subflow and one MSS from the second. This is equal to the increase of the fastest single-flow connection: one MSS per 50 ms.

By applying m to all subflows, we adapt the growth rate across all paths. This means that, although we favor the subflow which operates over the fastest path, we do not neglect the slower paths. Therefore, NMCC does not require probing to detect load changes on an unused path, whereas the coupled MPTCP algorithm introduces a special parameter for controlling the amount of probing. NMCC can therefore perform efficiently in heterogeneous environments, adapting fast to path failures and congestion bursts. For instance,

consider an integrated terrestrial-satellite network where the terrestrial link has 10 ms delay and the satellite one 250 ms. In this case $m = 1.004$, hence window growth is not constrained and NMCC effectively grasps the available capacity.

2) Slow Start: Most work on multiflow transport only deals with congestion avoidance, disregarding slow start. Nevertheless, during the evaluation of NMCC we noticed that friendliness was compromised when (i) the content was relatively small and (ii) the path was very congested. An analysis of the evolution of the congestion windows showed that NMCC gained bandwidth almost N-times faster than a single-flow connection during slow start, with N subflows. Since short and very congested connections spend a substantial fraction of their lifetimes in slow start, meeting the friendliness goals in congestion avoidance was not enough to amortize NMCC's aggressive behavior during slow start.

One way to reduce aggressiveness during slow start is to reduce $ssthresh$ so as to move faster to congestion avoidance. Unfortunately, this has two disadvantages. First, when a connection starts, the available bandwidth of the communication path is unknown, thus $ssthresh$ should be set high enough to probe it. Second, reducing $ssthresh$ only limits the amount of bandwidth that the protocol will re-acquire, not its rate of acquisition. For this reason, we reused the friendliness factor m to also control slow start.

During slow start, a subflow i doubles its congestion window during a period of RTT_i; its growth rate is $\frac{cwnd_i}{RTT_i}$, while during congestion avoidance it drops to $\frac{1}{RTT_i}$. We introduce Ω_i and Ω'_i, the regular and the friendly growth rate of subflow i, respectively, where $\Omega_i = m * \Omega'_i$. Again, we want to match the growth rate of the most aggressive flow, Ω_{max}, therefore we have the following equation:

$$\Omega_{max} = \sum_{i=1}^{N} \Omega'_i = \sum_{i=1}^{N} \frac{\Omega_i}{m}$$

for N jointly controlled subflows. We can then calculate m based on the regular growth rates of all subflows as follows:

$$m = \frac{\sum_{i=1}^{N} \Omega_i}{\Omega_{max}}$$

Consequently, each flow's growth rate Ω'_i becomes $\frac{cwnd_i^{tcp}}{m*RTT_i}$ during slow start and $\frac{1}{m*RTT_i}$ during congestion avoidance, where $cwnd_i^{tcp}$ is the equilibrium window of TCP for path i. As increases in slow start are multiplicative, any change in window growth affects the subsequent increases: smaller windows grow slower. Therefore, during slow-start we use $cwnd^{tcp}$ in order to assure that the cumulative growth of NMCC is equal to single-flow TCP. Algorithm 1 provides the combined slow start and congestion avoidance algorithm. Note that the algorithm translates the "inflated RTTs" of NMCC into MPTCP-like "decreased window growths" to avoid any side-effects of prolonged timeouts, such as delayed loss detection.

Algorithm 1 Window adjustment and estimation of m.

```
1: procedure INCREASE_WINDOW
2:     if (cwnd < ssthresh) then
3:         cwnd ← cwnd + cwnd^tcp * MSS/(cwnd * m)
4:     else
5:         cwnd ← cwnd + MSS/(cwnd * m)
6:     end if
7: end procedure
1: procedure ESTIMATE_M
2:     max_rate ← 0
3:     total_rate ← 0
4:     for (i ∈ subflows) do
5:         if (subflow_state_i == CONG_AVOID) then
6:             rate ← MSS/RTT_i
7:         else
8:             rate ← cwnd_i/RTT_i
9:         end if
10:        total_rate ← total_rate + rate
11:        if (rate > max_rate) then
12:            max_rate ← rate
13:        end if
14:    end for
15:    m ← total_rate/ max_rate
16: end procedure
```

V. Performance Evaluation

In this section we focus on the extent to which our hybrid congestion control can meet the friendliness requirement of multiflow transfers in different network scenarios. We have implemented our scheme as part of the mmTP protocol that runs over Blackadder, the PSI prototype implementation [21]. Our implementation includes the mmTP sender and receiver applications with NMCC enabled, as well as a TM that computes the k-shortest paths from every publisher to a subscriber, using the algorithm by Yen [22] with hop count as the metric.[2]

We deployed Blackadder with mmTP in several LAN topologies, using 100 Mbit switches and workstations as network nodes. Our experiments examine (i) the effect of TM assistance when paths are disjoint, (ii) the effectiveness of NMCC with overlapping paths, (iii) NMCC's behavior in short transfers (iv) the friendliness of NMCC and coupled MPTCP and (v) NMCC's behavior in heterogeneous networks.

In our testbed, the transmission latency among all nodes is 0.2-0.3 ms and the bandwidth of each link is 11.7 MB/s, as estimated using `iperf`.[3] The duration of transfers during all experiments is 20 seconds, except when mentioned otherwise. In order to enhance the reliability of our conclusions, we repeated each experiment until the margin of error was less than 1%, so as to achieve a confidence level of 95%.

A. Disjoint paths

We first deployed mmTP in the topology of Figure 2(a), where we investigated the performance gains of our hybrid congestion control scheme when paths are known to be

[2]Our implementation is available at `http://mm.aueb.gr/`.

[3]Available at `http://iperf.sourceforge.net/`.

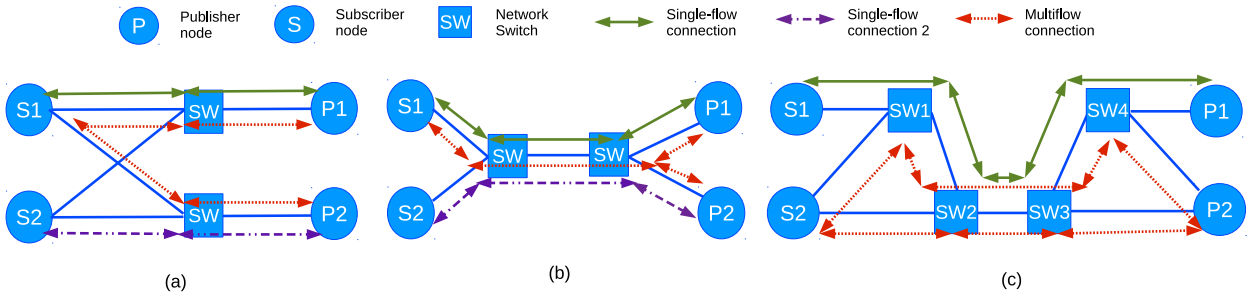


Fig. 2. Topology for performance evaluation with (a) disjoint paths and (b)-(c) shared paths.

Transmission mode	Transfer rate (MB/s)
Multipath with TM assistance	21.3
Multipath with no TM assistance	20.7
Single-flow from P1 to S1	10.6
Single-flow from P2 to S2	10.7
Single-flows on both paths	21.1

TABLE I
AVERAGE TRANSFER RATES WITH DISJOINT PATHS.

disjoint. Figure 2(a) supports one multisource path from publishers P1 and P2 to subscriber S1 and two single-paths from publishers P1 and P2 to subscribers S1 and S2, respectively. Thereupon, we ran some experiments with no contending traffic, so as to establish a performance baseline, leading to the average transfer rates shown in Table I. These experiments include deployment of multiflow mmTP connections with and without TM assistance, as well as single-flow mmTP connections. We notice that each path offers roughly 10.6 MB/s throughput and multiflow mmTP achieves 21.3 and 20.7 MB/s with and without TM assistance, respectively. These preliminary results validate that mmTP fully exploits available capacity and imply that TM assistance slightly enhances performance, even in the absence of competitive flows.

We then deployed mmTP in multipath mode over the same topology (S1 requests data from both P1 and P2), with one or two single-flow connections competing over one or both disjoint paths (S1 to P1 and S2 to P2). In Figure 3(a) we show the average share of the total bandwidth that mmTP achieved in each case, depending on whether TM assistance was turned on or off. The results validate the performance gains and the friendliness of NMCC. Ideally, with one contending single-flow connection NMCC should use half of the bandwidth over one path and the entire bandwidth over the other, or 75% of the total bandwidth, while with two contending single-flow connections NMCC should use half of the bandwidth over each path, or 50% of the total bandwidth. With TM assistance, mmTP acquires 67.5% and 49.5% of the overall bandwidth, respectively. Not only is this higher than with no TM assistance, it is also closer to the ideal bandwidth share. The bandwidth shares of mmTP with no TM assistance, which are only 52.6% and 36.8%, respectively, correspond to an equal share of the bandwidth among all connections, disregarding the actual topology.

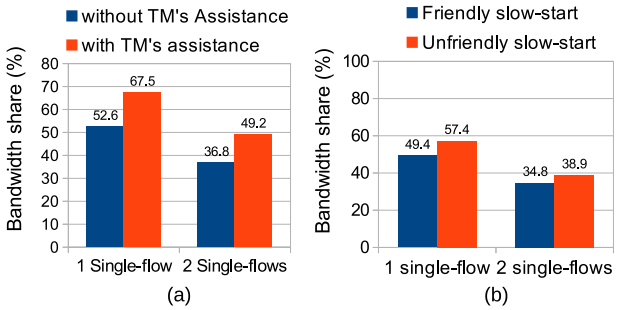


Fig. 3. Bandwidth share of mmTP (a) with and without TM assistance and (b) with and without friendly slow start in short transfers.

B. *Shared paths*

To investigate the case where paths share some links, mandating a less aggresive behavior to ensure friendliness, we used the topology shown in Fig. 2(b), where Publishers and Subscribers are connected by paths sharing a link. We deployed a multisource connection from subscriber S1 to publishers P1 and P2, in parallel with 1, 2, 4 and 9 single-flow connections from subscriber S1 to publisher P1 and from subscriber S2 to publisher P2; these connections are distributed uniformly between the two paths.

Fig. 4(a) demonstrates the average bandwidth percentage acquired by NMCC and *all* single-flow connections, while Fig. 4(b) displays the average transfer rate achieved by NMCC and the *average* unicast connection. NMCC acquires 51.1%, 35.5%, 21.5% and 10.8% of the bottleneck link's bandwidth when competing with 1, 2, 4 and 9 single-flow connections, respectively, marginally over the optimal sharing ratios of 50%, 33.3%, 20% and 10%, respectively, thus satisfying the friendliness goal. The slight performance advantage of NMCC, also evident in the transfer rates, is a side effect of the friendliness constraint: since window growth is distributed across all subflows, NMCC approaches congestion limits gradually, resulting in slightly less retransmissions than the average single-flow connection (2.1% on average).

We also examined NMCC's response to a sudden change in the congestion level, by repeating the previous experiment, but this time starting the multiflow connection either 7 sec after or 7 sec before the start of the single-flow connections.

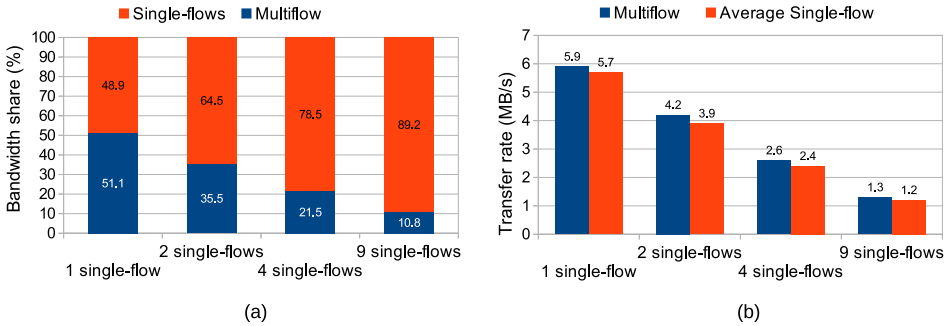


Fig. 4. (a) Bandwidth shares of NMCC and all single-flow connections. (b) Transfer rate of NMCC and the average single-flow connection.

The results of these experiments are nearly identical to the previous ones, as NMCC acquires 52%, 35.6%, 21.1% and 10.8% of the bandwidth when competing with 1, 2, 4 and 9 single-flow connections, respectively. Consequently, NMCC manages to efficiently share bandwidth with newly established connections, as well as to obtain a fair share of bandwidth when launched in an already congested path.

C. Short Transfers

NMCC is friendly during slow-start, unlike MPTCP which is only concerned with congestion avoidance. This is particularly important for short transfers, where friendly congestion avoidance cannot compensate for an unfriendly slow start. To evaluate this aspect of NMCC, we reused the shared link topology of Fig. 2(b), deploying one multisource NMCC connection and either 1 or 2 contending single-flow connections. Each connection transfers a 10 MByte object, which would require less than 1.1 second to complete in the absence of contention. Fig. 3(b) presents the percentage of overall bandwidth acquired by NMCC when friendly slow start is turned on or off.

With unfriendly slow start, NMCC grabs a disproportionate amount of bandwidth from the competing connections, compared to the ideal shares of 50% and 33%. In the first case, NMCC gets 57.4% of the bandwidth; while in the second case it gets 38.9%, or 14.8% and 16.8% more than the fair share, respectively. On the other hand, NMCC with friendly slow start gains 49.4% and 34.8% of the total bandwidth. Consequently, NMCC is friendly even with short transfers.

For even shorter transfers, for example Web objects a few KBytes long, the unfairness is even more pronounced, as such connections can easily complete during slow start. The reason for presenting results from a 10 MByte transfer is to show that the initial over-aggressiveness during slow start cannot be compensated even with longer transfers.

D. Friendliness of NMCC and MPTCP

We then compared the friendliness of the hybrid approach of NMCC and the coupled congestion control of MPTCP [2]. MPTCP's design is similar to NMCC, in that congestion management takes place at the endpoints and time-out estimation is based on RTTs. These similarities simplified the implementation of the coupled congestion control algorithm of MPTCP in our mmTP implementation.

For these experiments we used the topology of Fig. 2(c), where all paths share at least one link; MPTCP's inability to support TM assistance would make a comparison over disjoint paths unfair. We deployed a number of single-path flows from subscriber S1 to publisher P1, as well as multipath flows from subscriber S2 to publisher P2, using the paths indicated in Fig. 2(c). Multipath connections utilized either the coupled MPTCP or the NMCC algorithm. We denote each experiment as $X : Y : Z$, where X shows the number of single-path flows, Y shows the number of multipath flows using coupled MPTCP and Z shows the number of those using NMCC.

The results of these experiments are summarized in Fig. 5. Fig. 5(a) displays the deviation of the obtained bandwidth of each connection from its fair share which, due to the shared link, is given by $\frac{Link\_Capacity}{\#Connections}$. Results below 0% indicate overly friendly flows, while results over 0% indicate overly aggressive ones. We can distinguish three groups in this figure. The first group reflects experiments '1:3:3', '1:2:2' and '1:1:1', where MPTCP is too friendly, resulting in poor performance. The second group reflects experiment '2:1:1', where all connections are close to their fair shares. The third group reflects experiments from '3:1:1' to '8:1:1', where multiflow connections are more aggressive, making single-flow ones lose some of their share.

Figure 5(b) presents the above results for multipath connections normalized to the bandwidth achieved by single-path flows, that is, we divide the bandwidth share obtained by MPTCP and NMCC by the bandwidth achieved by the average single-flow connection. Thereupon, the closer the score is to 1 the friendlier a connection is to single-flow. Based on this figure we can argue that NMCC is more friendly to single-flow connections than MPTCP most of the time. Even though the performance superiority of NMCC is mostly evident when there are fewer single-path flows competing for capacity, we observe that NMCC gives more consistent results in general.

E. Heterogeneous Networks

Finally, we explored NMCC's performance in heterogeneous networks where paths exhibit diverse capacity, delays

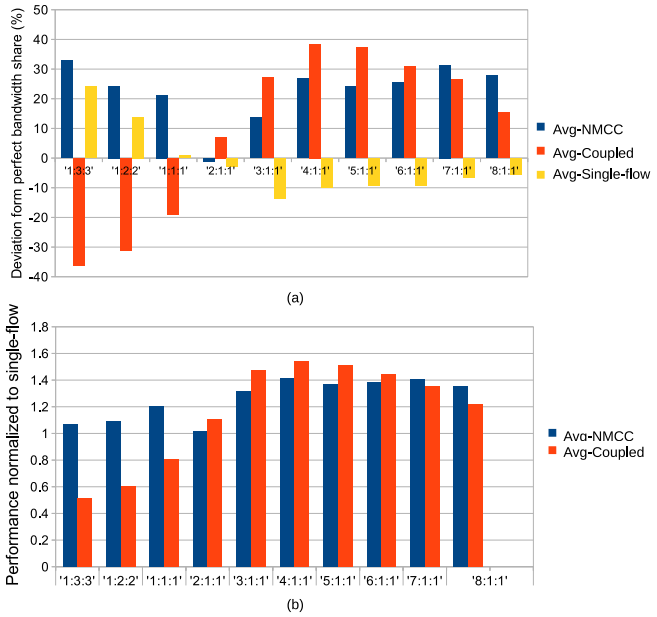


Fig. 5. (a) Deviation of obtained bandwidth from fair shares, (b) Difference of multiflow and single-flow deviation from fair shares normalized to the single-flow deviation from fair shares.

and error-rates. In order to emulate these conditions, we replicated the RTT-mismatch scenario used in the evaluation of coupled MPTCP [2]. This scenario assumes a smartphone device that uses simultaneously two disjoint paths: (a) a WiFi link with 10 ms delay and 4% error-rate and (b) a 3G link with 100 ms delay and 1% error-rate. First, we used netem[4] to configure the delay and error-rate of the multisource paths in the topology of Figure 2(a) and then we deployed mmTP with no contending traffic, so as to study window growth without congestion. We investigated the behavior of NMCC against both the coupled and uncoupled MPTCP congestion control algorithms. Figure 6 presents the number of packets that are sent over the WiFi link within a period of 60 seconds; we neglect the 3G link, as it is identically saturated by all algorithms. The results validate the expected performance superiority of NMCC. The significant RTT divergence leads NMCC to compute a low friendliness factor ($m \simeq 1.1$) which offers similar performance to the uncoupled MPTCP algorithm, thus grasping all available capacity from the start. In contrast, coupled MPTCP fails to adapt to this RTT mismatch, as it utilizes less than 93% of the available capacity until 10 sec and roughly 96% thereon.

VI. In-network assistance in IP networks

Our hybrid congestion control mechanism for multiflow transfers, NMCC, relies on an in-network scheme that reports shared bottlenecks to the end-hosts. The PSI architecture is an appropriate terrain for this design, since it provides a TM function that (i) is aware of network topology and (ii) interacts

[4]http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

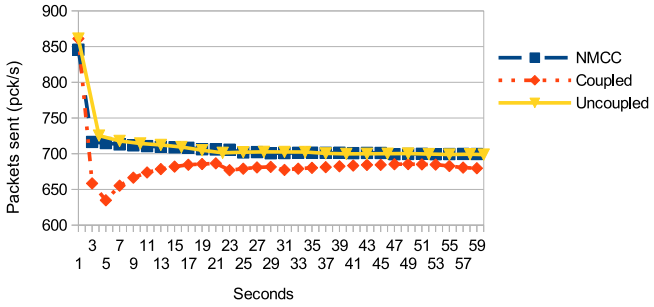


Fig. 6. Packets/sec sent over the WiFi link for a 60 sec period.

with the end-hosts. The TM knows the physical structure of the network, so it can easily detect shared bottlenecks. In addition, when two pub/sub requests are matched, the TM sends the LIPSIN identifiers of the paths directly to the applications, therefore it directly pushes the topological information to the users. In order to extend our scheme to other types of networks, such as IP-based ones, we need equivalent in-network mechanisms to provide such information.

A technology that offers centralized path selection and source routing in IP networks is *Multi-Protocol Label Switching* (MPLS) [23]. MPLS is used in backbone networks, where it applies QoS-based traffic control by classifying flows and forwarding them via predefined routes. Short fixed-length labels are assigned to packets at the ingress to an MPLS cloud, and these labels are used to make forwarding decisions inside the MPLS domain. The path formation process is generic, allowing route computation by the underlying routing protocols or explicit definition by a network operator. Multipath deliveries are also encouraged, in the form of splitting single-flow connections into several subflows at the ingress router.

Currently, MPLS is used for applying domain-scale traffic engineering rather than for enhancing the performance of individual connections, hence, connection splitting is done with static sharing weights for general load balancing. Consequently, congestion control takes place at the actual end-hosts (i.e the users), while the ingress MPLS router is confined to the flow control of the available paths. However, if we consider the ingress router as the congestion manager of the MPLS cloud, as it splits the flow, assigns labels to each of its subflows and becomes the end-host of a local MPLS service, then our network-assisted congestion control can be integrated to the MPLS network. Specifically, when the network operator discovers multiple paths for bulk flows and sends the corresponding labels to the ingress router, it also sends information on how flows are grouped depending on path sharing, as described in Sec. IV-A. The ingress router, which runs NMCC for each bulk flow, exploits this information and source routing to selectively engage the friendliness mechanism.

Software-Defined Networking (SDN) [24] is a novel networking scheme that can be used to achieve similar goals to PSI, including centralized path selection. In SDN, pro-

grammable switches forward packets based on "dynamic" rules that bind flow identifiers, such as fields of the IP header, with outgoing network interfaces. These rules are defined by a centralized controller that is aware of the network topology and forms virtual circuits by explicitly sending rules to all on-path routers. Circuit creation can be reactive, where a router ask the controller's assistance when no rule can be applied to a received packet, or proactive, where the controller forms the route a priori, for example, to achieve load balancing. In both cases, SDN operation is transparent to the end-hosts that manage congestion control.

As the SDN controller does not communicate with end hosts, which means that it cannot pass any topological information to them, we can apply the same ideas as for MPLS to introduce in-network assistance and NMCC to SDN clouds, by considering the ingress SDN router as the congestion manager of bulk flows. When the SDN controller creates forwarding paths by sending the appropriate rules to the SDN switches, it can send information on how flows are grouped depending on path sharing to the ingress SDN router, as well as instructions on how to tag each IP header so as to implicitly select the appropriate path. The ingress SDN router will then run NMCC for each bulk flow, as above.

Adding in-network assistance to MPLS or SDN clouds may raise two concerns: (i) the computational costs of applying congestion control for numerous flows at the ingress router may degrade scalability, (ii) the limited application scope of backbone networks may prevent fully exploiting all connectivity options. For example, when multihomed devices connect to different access networks, these may not employ the same MPLS or SDN cloud, preventing the transparent use of NMCC within each separate cloud.

VII. Conclusions

We presented a hybrid congestion control algorithm for multiflow transport, consisting of NMCC and an in-network assistance mechanism. Our design offers friendliness to single path connections using TCP-like congestion control, while increasing the utilization of network resources. It achieves this by detecting shared physical bottlenecks and managing aggressiveness appropriately, without requiring complex network signaling or adding state to routers. We have implemented the congestion control algorithm in the PSI architecture prototype and evaluated its performance gains in several topological and traffic scenarios. Our results not only verify the effectiveness of our design, they also validate its performance superiority over MPTCP's coupled congestion control algorithm in short transfers and heterogeneous networks. Finally, we discussed how in-network assistance can be provided in IP networks based on centralized routing, such as MPLS or SDN.

Acknowledgement

The work presented in this paper was supported by the EU funded H2020 ICT project POINT, under contract 643990.

References

[1] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies." *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[2] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. of the USENIX NSDI Conference*, 2011.

[3] C. Hopps, "Analysis of an equal-cost multi-path algorithm," RFC 2992, 2000.

[4] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, no. 3, pp. 28–37, May 2001.

[5] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. of the PLFDNeT Workshop*, 2009.

[6] P. Key, L. Massoulie, and D. Towsley, "Path selection and multipath congestion control," in *Proc. of the IEEE INFOCOM*, 2007, pp. 143–151.

[7] M. Becke, T. Dreibholz, H. Adhari, and E. P. Rathgeb, "On the fairness of transport protocols in a multi-path environment," in *Proc. of the IEEE ICC*, 2012, pp. 2666–2672.

[8] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[9] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, "LIPSIN: line speed publish/subscribe internetworking," in *Proc. of the ACM SIGCOMM*, 2009, pp. 195–206.

[10] Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Multisource and multipath file transfers through publish-subscribe internetworking," in *Proc. of the ACM SIGCOMM ICN Workshop (poster session)*, 2013, pp. 43–44.

[11] ——, "Accelerating file downloads in publish subscribe internetworking with multisource and multipath transfers," in *Proc. of the World Telecommunications Congress (WTC)*, 2014, pp. 1–6.

[12] M. Zhang, J. Lai, A. Krishnamurthy, L. L. Peterson, and R. Y. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths." in *Proc. of the USENIX Annual Technical Conference*, 2004, pp. 99–112.

[13] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 381–395, 2002.

[14] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, "Multipath congestion control in content-centric networks," in *Proc. of the IEEE INFOCOM NOMEN Workshop*. IEEE, 2013, pp. 363–368.

[15] S. Oueslati, J. Roberts, and N. Sbihi, "Flow-aware traffic control for a content-centric network," in *Proc. of the IEEE INFOCOM*, 2012, pp. 2417–2425.

[16] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an information-centric internetworking architecture," *SIGCOMM Computer Communications Review*, vol. 40, no. 2, pp. 26–33, Apr. 2010.

[17] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. Siris, and G. Polyzos, "Caching and mobility support in a publish-subscribe Internet architecture," *IEEE Communications*, vol. 50, no. 7, pp. 52–58, July 2012.

[18] B. A. Alzahrani, M. J. Reed, J. Riihijärvi, and V. G. Vassilakis, "Scalability of information centric networking using mediated topology management," *Journal of Network and Computer Applications*, 2014.

[19] Y. Thomas, P. A. Frangoudis, and G. C. Polyzos, "Qos-driven multipath routing for on-demand video streaming in a publish-subscribe internet," in *Proc. of the IEEE Workshop on Multimedia Streaming in Information-centric Network (MuSIC)*. IEEE, 2015, pp. 1–6.

[20] T. Henderson, E. Sahouria, S. McCanne, and R. Katz, "On improving the fairness of TCP congestion avoidance," in *Proc. of the IEEE GLOBECOM*, vol. 1, 1998, pp. 539–544 vol.1.

[21] G. Parisis, D. Trossen, and D. Syrivelis, "Implementation and evaluation of an information-centric network," in *Proc. of the IFIP Networking Conference*, 2013, pp. 1–9.

[22] J. Yen, "Finding the k shortest loopless paths in a network," *Science Management*, vol. 17, no. 11, pp. 712–716, 1971.

[23] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over mpls," RFC 2702, 1999.

[24] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications*, vol. 51, no. 2, pp. 114–119, 2013.

Characterizing Interest Aggregation in Content-Centric Networks

Ali Dabirmoghaddam* Mostafa Dehghan† J. J. Garcia-Luna-Aceves*‡
*University of California Santa Cruz, †University of Massachusetts Amherst, ‡PARC
{alid, jj}@soe.ucsc.edu, mdehghan@cs.umass.edu

***Abstract*—The Named Data Networking (NDN) and Content-Centric Networking (CCN) architectures advocate Interest aggregation as a means to reduce end-to-end latency and bandwidth consumption. To enable these benefits, Interest aggregation must be realized through Pending Interest Tables (PIT) that grow in size at the rate of incoming Interests to an extent that may eventually defeat their original purpose. A thorough analysis is provided of the Interest aggregation mechanism using mathematical arguments backed by extensive discrete-event simulation results. We present a simple yet accurate analytical framework for characterizing Interest aggregation in a CCN router, and use our model to develop an iterative algorithm to analyze the benefits of Interest aggregation in a network of interconnected routers. Our findings reveal that, under realistic assumptions, an insignificant fraction of Interests in the system benefit from aggregation, compromising the effectiveness of using PITs as an integral component of Content-Centric Networks.**

I. Introduction

The fact that the content and not its location is what matters to end-users has given rise to many recent proposals classified under the generic name of Information-Centric Networking (ICN) [1]. Many ICN blueprints can be seen as *Interest-driven* communication models, where users ask for content by name through *Interest* packets. The most prominent examples of Interest-driven ICN (NDN [2] and CCNx [3]) are based on three main components. Routers maintain *Forwarding Information Bases* (FIB) listing routes to name prefixes, which enable routing to *names* rather than addresses. Given that all copies of the same content are equivalent, routers cache content opportunistically in their local *Content Store* (CS). In addition, each router maintains a *Pending Interest Table* (PIT) to suppress unnecessary Interests.

When a router receives an Interest that cannot be satisfied through its local CS, it creates a PIT entry and forwards the Interest if there is no entry for the same content name in its PIT. If a PIT entry already exists for the content name, the Interest is *aggregated* at the router and is not forwarded. The idea of Interest aggregation is hardly new. It has been implemented in Web caching architectures in the past, *e.g.*, Squid [4]—where it was referred to as *collapsed forwarding*—and commercially used on production content delivery networks since the early days of the Web.

The expectation of Interest aggregation in ICN architectures has been that network and server loads can be drastically reduced by suppressing similar Interests, and that end-to-end latencies can be reduced by integrating caching with Interest aggregation. These benefits, however, come at a non-trivial cost. Creating and maintaining the PIT is expensive, especially when performed at Internet scale. The routers used in the Internet backbone handle hundreds of thousands of packets every second. The proposed data structure must be fast enough when operating at its peak capacity to not act as a source of latency and overhead itself. Thus, considerable work has focused on optimization and scalability of the PIT (*e.g.*, see [5]–[8]).

We are not aware of any comprehensive analytical work characterizing the expected benefits of Interest aggregation. Some experimental efforts [9], [10] have been made in understanding the dynamics of the PIT size; however, important questions such as *what fraction of Interests are subject to aggregation under realistic conditions* and *whether or not that justifies the use of PITs* have remained unanswered to this date.

To answer these questions, Section II presents a simple yet powerful analytical toolbox for characterizing a CCN router with a CS and a PIT. We compute the cache hit probability at the CS, the Interest aggregation probability at the PIT, as well as the router response time at an object-level granularity. Section III uses these constructions for the analysis of a network of interconnected content routers. Through extensive event-driven simulations, Section IV demonstrates how accurately our proposed framework can predict the steady-state behavior of such a complex system. Furthermore, it shows how the model can be used to study the performance of a large network under realistic conditions, such as that of today's Internet, for which event-driven simulations are prohibitive.

Numerical evaluations of our model for large-scale systems reveal that only a small fraction of Interests may actually benefit from Interest aggregation in realistic settings. These benefits are highly dependent on the amount of caching budget available to the network. For example, a 5% cumulative aggregation percentage can be achieved using a per-node caching capacity of equal to 0.05% of the total number of objects in the system, while increasing the budget to just 0.5% reduces the aggregation percentage to below 1%. Even worse, our findings show that most Interest aggregation takes place closer to where the content is permanently stored—*i.e.*, near the producers deep in the core of the network—where aggregating Interests hardly makes sense anymore.

The insights from our modeling results lead to the necessary conclusion that Interest aggregation should *not* be an integral

ISBN 978-3-901882-83-8

component of future ICN architectures and Content-Centric Networks. On-path caching or edge caching provide all the benefits of reducing the number of Interests that request similar content, without the costs of maintaining PITs. In turn, if per-Interest forwarding state is not needed for other reasons, this realization makes Content-Centric Networking at Internet scale more feasible than the current NDN design, given that forwarding data structures (*e.g.*, CCN-DART [11], [12] and CCN-GRAM [13]) smaller and more efficient than PIT could be used for routing data back to the consumers.

II. CCN Router with Non-zero Download Delays

We develop a mathematical model to characterize a CCN router with a Content Store (CS) to enable caching functionality and a Pending Interest Table (PIT) that allows Interest aggregation. Unlike previous work [14]–[17], we assume that the content download delays are non-zero. Our derivations, hence, can be regarded as an extension to a highly accurate approximation of LRU cache introduced by Che *et al.* [15].

Consider a content router with a CS of capacity C implementing LRU replacement policy receiving Interests indexed (without loss of generality) in their decreasing order of popularity from 1 to N. In the rest of this paper, the terms CS and cache refer to the same concept and we shall use them interchangeably. Assume that Interests conform to the Independent Reference Model (IRM), *i.e.*, for every object, Interests inter-arrival times to the router are independent, identically distributed (i.i.d.) random variables. Fig. 1 illustrates Interests (shown as combs) as arriving at a content router over time. We focus on the Interests for an object i, which are highlighted in color. In Fig. 1, the red and green zones respectively specify time intervals when object i is absent or present in the CS. Upon receiving an Interest, if the CS contains a copy of object i, a *cache hit* occurs (see green combs); the Interest is immediately satisfied and a copy of that object is sent back to the requester. Otherwise, we say a *cache miss* occurs (see red combs).

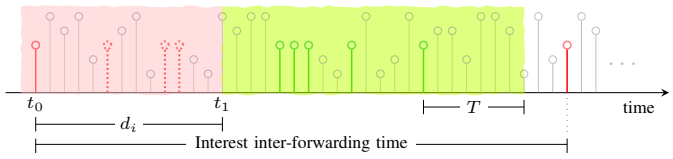


Fig. 1: Interests arriving at a content router over time

Let t_0 be the instant when the first cache miss for object i occurs at the CS. At that point, the content router creates a PIT entry for that object and forwards the Interest to another router/source (forwarded Interests shown as solid red combs). Let d_i be the random variable indicating the duration till a copy of the requested object is downloaded and stored in the CS, and mark that instant as $t_1 = t_0 + d_i$. We shall refer to d_i as the *download delay* of object i. Any subsequent Interest for object i during the interval (t_0, t_1) is aggregated at the content router due to the existence of a PIT entry under object i's name (aggregated Interests shown as dotted red combs).

A set of events take place at time t_1, namely the content router: (1) stores a copy of object i in the local CS; (2) removes the PIT entry for the corresponding Interest, and (3) forwards a copy of object i on all interfaces from which a request for it had been received. A copy of object i remains in the CS so long as the inter-arrival time of consecutive Interests for object i is smaller than T denoting the *characteristic time* of the cache introduced by Che *et al.* [15] (see Fig. 1). In essence, T is a random variable signifying the duration it takes until C distinct objects other than i are downloaded into the cache and object i is dismissed. For relatively large C and N and when the content download delays into the cache are zero, Fricker *et al.* [18] proved that T becomes deterministic. Later Dehghan *et al.* [19] proved this right even when download delays are non-zero.

The characteristic time T depends on the cache capacity C, the Interests arrival rate and the object popularity distribution and is computed according to

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] = C\,, \tag{1}$$

where X_i is the Bernoulli random variable indicating whether object i is present in the cache or not. Eq. (1) comes from the fact that the cache has the capacity for C objects. Note that $\mathbb{E}[X_i]$ equals the probability of object i being present in the cache, *i.e.*, the cache occupancy probability. With Poisson arrivals and thanks to the PASTA property, the cache occupancy probability equals the cache hit probability for any object i. Hence, we arrive at

$$\sum_{i=1}^{N} h_i = C\,. \tag{2}$$

where h_i denotes the cache hit probability of object i. We shall later use (2) as a constraint in computing the cache hit probability of individual objects.

A. Computing the Cache Hit Probability

Under the assumption that request inter-arrival times are independent exponential random variables, for a particular object i, the p.m.f. of exactly $n_i = k$ cache hits is the probability of the event that the first k Interest inter-arrival times are smaller than T, while the following Interest inter-arrival time is greater than T. This probability can be formalized by the geometric distribution $\mathbb{P}\left(n_i = k\right) = \left(1 - e^{-\lambda_i T}\right)^k e^{-\lambda_i T}$, and the expected number of cache hits is derived as

$$\mathbb{E}[n_i] = \sum_{k=0}^{\infty} k\,\mathbb{P}\left(n_i = k\right) = e^{\lambda_i T} - 1\,.$$

After forwarding a missed request for object i, if $\mathbb{E}[d_i]$ denotes the expected time to download a copy of i into the CS, the expected number of missed requests during this interval would be $\mathbb{E}[\bar{n}_i] = 1 + \lambda_i \mathbb{E}[d_i]$, of which one Interest is forwarded and the remaining $\lambda_i \mathbb{E}[d_i]$ are aggregated at the PIT. The expected total number of Interests received for object i during one such inter-forwarding cycle is then $\mathbb{E}[N_i] = \mathbb{E}[n_i] + \mathbb{E}[\bar{n}_i]$.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Consequently, the probability of a cache hit for object i is derived as

$$h_i = \frac{\mathbb{E}[n_i]}{\mathbb{E}[N_i]} = \frac{e^{\lambda_i T} - 1}{\lambda_i \mathbb{E}[d_i] + e^{\lambda_i T}} \,. \tag{3}$$

Eq. (3) can be regarded as an extension to the LRU approximation of Che *et al.* [15] where download delays can be non-zero. In fact, setting download delays to zero simplifies Eq. (3) to their renown form of $h_i = 1 - \exp(-\lambda_i T)$.

B. Computing the Interest Aggregation Probability

We now turn to computing the probability of Interest aggregation at the PIT. From our previous discussion, the expected number of aggregated requests during the download interval $\mathbb{E}[d_i]$ are $\mathbb{E}[\bar{n}_i] - 1 = \lambda_i \mathbb{E}[d_i]$. The probability of an Interest for object i being aggregated at the PIT is subsequently derived as

$$a_i = \frac{\mathbb{E}[\bar{n}_i] - 1}{\mathbb{E}[N_i]} = \frac{\lambda_i \mathbb{E}[d_i]}{\lambda_i \mathbb{E}[d_i] + e^{\lambda_i T}} \,. \tag{4}$$

Put differently, Eq. (4) states what fraction of Interests for object i arriving at the content router are aggregated in the long run.

C. Computing the Router Response Time

Another important measure in the analysis of interconnected routers when the download delays are non-zero is the router response time. Due to the potential existence of a PIT entry in the content router when an Interest for object i is received, the time it takes the router to satisfy that Interest can be any value from interval $(0, d_i]$.

We define the *pending time* of an Interest in the PIT as the time difference between the arrival of the Interest to the router and the subsequent moment when the Interest is served. With Poisson arrivals, Interest arrival times are uniformly distributed over $(0, d_i]$; hence, the sum W_i of pending time of Interests during a download interval d_i can be formulated as

$$W_i = d_i + \lambda_i \int_0^{d_i} (d_i - t)\, \mathrm{d}t = d_i(1 + 0.5\lambda_i d_i) \,.$$

We define the response time r_i of the content router for a particular object i as the expected pending time of Interests for that object which is readily derived as

$$r_i = \frac{\mathbb{E}[W_i]}{\mathbb{E}[N_i]} = \frac{\mathbb{E}[d_i(1 + 0.5\lambda_i d_i)]}{\lambda_i \mathbb{E}[d_i] + e^{\lambda_i T}} \,. \tag{5}$$

The router response time depends on the distribution of download delays, though knowledge of only the first two moments is sufficient.

III. An Algorithm for the Analysis of Hierarchical CCN Networks

We investigate how an interconnected network of CCN routers can be analyzed using the results from the previous section. Consider a hierarchy of routers as depicted in Fig. 2. Consumers are located at the bottom level where their requests for objects of interest are directed to the first level CCN routers (*i.e.*, ℓ_1 routers). An ℓ_1 router searches its local CS for a copy of the requested object and if failed, it forwards the request to the next level router (*i.e.*, the parent ℓ_2 router). This process is repeated on every cache miss and in the worst case, the requested object is downloaded directly from the producer at the top of the hierarchy storing permanent copies of all the objects in the system. On the reverse path back to the original requester, a copy of the object is stored in the CS of every CCN router it passes through.

For simplicity, we consider only a single producer in the network. The producer in our model can alternatively be conceived as a collection of several producers at the core of the network collapsed into one single entity. This single-source spanning-tree simplification of the network topology is standard in many studies of content delivery [20] and publish-subscribe networks [21].

There are two important challenges in the analysis of the foregoing structure. First, the Interest stream into a higher-level router (*i.e.*, all except ℓ_1 routers) is no longer a simple Poisson process, but an aggregate of miss streams from a number of lower-level routers. It is known, however, that the superposition of multiple streams tends toward Poisson as the load increases [22], [23]. We shall use this insight when extending the results of the previous section to the analysis of CCN networks by primarily focusing on trees of higher arity.

Secondly, chaining routers may cause circular dependencies in the computation of some router performance metrics. For instance, the cache hit probability in an ℓ_1 router depends on the download delay of the objects as mandated by Eq. (3). That delay is determined by the response time of the parent ℓ_2 router which in turn is a function of its input rate. The input into an ℓ_2 router itself partially relies on the miss stream of its descendant ℓ_1 router, and that is how a dependency cycle is formed. To overcome this hurdle, we present an iterative approach as outlined in Algorithm 1.

Procedure ANALYZE-CCN-TREE is proposed to compute the important router performance metrics we discussed in Section II for a hierarchical network structure such as the one portrayed in Fig. 2. The procedure analyzes a complete k-ary tree of height $L + 2$ in which consumers are at level 0; L layers of CCN routers are employed in the middle, and the producer is located at level $L + 1$ as the root of the tree. The available caching budget is provided by vector

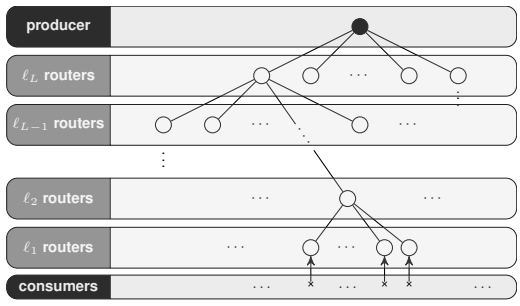


Fig. 2: A partial view of a hierarchy of interconnected routers

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Algorithm 1 Method to characterize a hierarchical CCN represented by a complete tree as depicted in Fig. 2

Input: k: arity of the tree; L: number of tree levels; λ: consumer input rate to each first level router; δ: round-trip delay across each link; $\boldsymbol{C}$: vector of caching budget per node per layer; $\boldsymbol{q}$: probability vector reflecting the object popularity profile.

Output: T: characteristic time of caches at each level; $\boldsymbol{h}$: vector of cache hit probabilities at each level; $\boldsymbol{a}$: vector of aggregation probabilities at each level; $\boldsymbol{r}$: vector of router response times at each level; $\boldsymbol{m}$: vector of incoming Interest rates to each level.

```
1  procedure ANALYZE-CCN-TREE(k, L, λ, δ, C, q)
2      i ← 0
3      for ℓ from 1 to L do
4          r_{ℓ+1}^{(i)} ← δ × (L − ℓ)
5      end for
6      while not converged do
7          i ← i + 1
8          for ℓ from 1 to L do
9              d_ℓ^{(i)} ← δ + r_{ℓ+1}^{(i−1)}
10         end for
11         m_1^{(i)} ← λ × q
12         for ℓ from 1 to L do
13             T_ℓ^{(i)}  ← CHAR-TIME(m_ℓ^{(i)}, d_ℓ^{(i)}, C_ℓ)
14             h_ℓ^{(i)}  ← HIT-PROB(m_ℓ^{(i)}, d_ℓ^{(i)}, T_ℓ^{(i)})
15             a_ℓ^{(i)}  ← AGG-PROB(m_ℓ^{(i)}, d_ℓ^{(i)}, T_ℓ^{(i)})
16             r_ℓ^{(i)}  ← RESP-TIME(m_ℓ^{(i)}, d_ℓ^{(i)}, T_ℓ^{(i)})
17             m_{ℓ+1}^{(i)} ← MISS-RATE(k, m_ℓ^{(i)}, h_ℓ^{(i)}, a_ℓ^{(i)})
18         end for
19     end while
20 end procedure
```

$\boldsymbol{C}$ in which element C_ℓ indicates the allocated CS capacity for each of the routers at level ℓ. The initial rate at which Interests are produced by the consumers and fed into each ℓ_1 router is λ. The object popularity profile follows a Zipfian distribution as determined by the probability vector $\boldsymbol{q}$. Without loss of generality, in this paper we always rank objects in their decreasing order of popularity. As such, the normalized popularity of the n^{th} ranked object is determined by the power-law $q(n) = n^{-\alpha}/\sum_{u=1}^{N} u^{-\alpha}$, where exponent $\alpha > 0$ is the parameter to the Zipf distribution. Finally, each link induces a round-trip delay of δ for transporting an individual content object. These parameters are inputs to Algorithm 1.

As pointed out, Algorithm 1 works in iterations to tackle circular dependencies. The superscript (i) used throughout the algorithm denotes the latest count of iterations. At the 0^{th} iteration, *i.e.*, the initial phase, since all caches are empty and all requests are fulfilled directly by the producer, the router response times (denoted by $\boldsymbol{r}$) are simply set based on the hop-distance of routers from the root (lines 3–5). The notation $\boldsymbol{r}_{\ell+1}^{(i)}$ describes the response time of a $(\ell+1)^{\text{th}}$ level router computed at the i^{th} iteration. Note that variables denoted in bold face are in fact vectors with values corresponding to individual objects in the system as ordered in popularity profile $\boldsymbol{q}$. Next, at any subsequent iteration:

1) Download delays for all levels are updated (lines 8–10) according to the heuristic that the delay for downloading files into the CS of an arbitrary router is equal to the response time of its parent router plus the round-trip delay of the link connecting them together. Assuming all objects are unit-sized, we can deduce that the average link delays are the same. In a hierarchical structure, thus, we can compute the download delays into a particular router by knowing the average link delays and the response time of the next level (*i.e.*, parent) router.
2) All performance measures discussed in Section II are computed/updated across all tree levels (lines 12–18).

Starting from the bottom working towards the top, at each tree level the measures are computed in the following order:

a) Procedure CHAR-TIME*:* is called at line 13 to compute the cache characteristic times by solving the following fixed-point equation for variable T:

$$\sum_{j=1}^{N} \frac{e^{m[j]T} - 1}{m[j]d[j] + e^{m[j]T}} = C\,, \tag{6}$$

where $m[j]$ and $d[j]$ are the j^{th} elements in vectors $\boldsymbol{m}$ and $\boldsymbol{d}$, respectively denoting the input rate and download delay for object j at the corresponding router. Note that Eq. (6) is indeed the expanded form of (2) using (3).

b) Procedures HIT-PROB*,* AGG-PROB *and* RESP-TIME*:* are called at lines 14–16 to use the above computed characteristic time for computing the cache hit probability, PIT aggregation probability and response time of routers for individual objects according to Eqs. (3), (4) and (5), respectively.

c) Procedure MISS-RATE*:* is called at line 17 to compute the aggregate miss rate *into* the next level (*i.e.*, parent) router using the above computed hit- and aggregation probabilities according to the following relation:

$$\boldsymbol{m}_{\ell+1} = k \cdot \boldsymbol{m}_\ell \odot (\boldsymbol{1} - \boldsymbol{h}_\ell) \odot (\boldsymbol{1} - \boldsymbol{a}_\ell)\,, \tag{7}$$

where $\odot$ signifies component-wise multiplication of corresponding vectors. In essence, Eq. (7) suggests that the input stream of a router at level-$(\ell+1)$ is the superposition of k miss streams from its descendant level-ℓ routers. The only exception are ℓ_1 routers whose input is directly provided by consumers according to line 11.

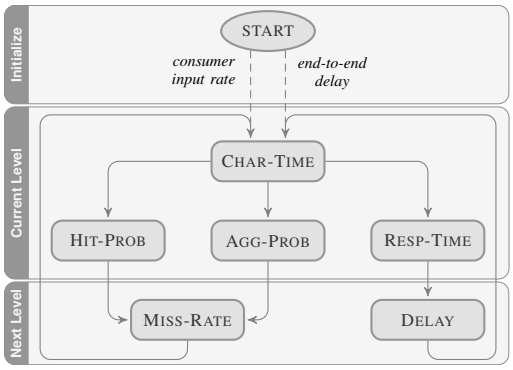


Fig. 3: Dependency among procedure calls in Algorithm 1.

To better understand the dependency between these procedure calls, the diagram in Fig. 3 provides a pictorial view

of their relationship. At the beginning, consumers' input rate and the end-to-end delay (between the consumers and the producer) are used to compute the cache characteristic time for all ℓ_1 routers. Cache hit- and PIT aggregation probabilities as well as router response times are then computed for ℓ_1 routers. Next these results are used to calculate the input rate and download delays for ℓ_2 routers. Then level 2 becomes the current level and a similar procedure is repeated for all remaining levels from the bottom to the top of the tree.

The computations in the middle and bottom boxes in Fig. 3 may be repeated in consecutive iterations as needed; the results from one iteration will be used in computing the next as the computed measures gradually converge to their steady-state values. In our numerical simulations—to be discussed next—we noticed that the first few iterations usually suffice to get an accuracy of better than 0.1% while no more than 10 iterations were needed in all cases studied (irrespective of the input size).

Implementing Algorithm 1 is straightforward in many off-the-shelf numerical computing environments. In our simulations, for solving Eq. (6), we leveraged `fsolve` function from MATLAB's Optimization Toolbox which uses trust-region methods [24] for solving systems of nonlinear equations. It is known [25] that trust-region methods take $\mathcal{O}(\epsilon^{-2})$ iterations to drive the norm of the gradient of the objective function below desired threshold ϵ. The time-complexity of Algorithm 1 is hence $\mathcal{O}(NL\,\epsilon^{-2})$.

IV. Performance Evaluation

We present simulation results to show how the proposed method can be used to accurately predict the complex behavior of a network of content routers. First, a detailed comparison of the numerical results of the presented model versus the results from extensive event-driven simulations in ndnSIM [26] is presented. Next, the results from our model are used to analyze more complex scenarios, such as networks with larger content base, which are far more cumbersome and time-consuming to study using conventional event-driven simulations.

We focus on two major strategies of cache allocation, namely *uniform caching* and *edge caching*. In the former, a fixed caching budget is evenly distributed across all content routers, whereas with the latter, the budget is entirely allocated to the routers at the edge of the network, *i.e.*, the ones directly serving the consumers. With edge caching, the upper level routers simply act as routers with no caching capability (that is, their CS size is set to zero). Yet they still perform Interest aggregation upon receiving Interests for which they have pending entries in their PITs.

A. Comparison of Model with Event-driven Simulations

We consider a tree of degree $k = 10$ and height $H = 5$ as the underlying topology, where $L = 3$ levels of content routers are used in the middle. The reason for using such a configuration is to keep the overall aggregate traffic pattern in the middle layers as close to being Poisson as possible, as discussed in Section III. Although the model was able to capture the overall trends in our experiments with trees of lower arity, we noticed that more accurate results are generally obtained when nodes have higher fan-in (*e.g.*, 10 or more). This assumption, however, is not unrealistic as some studies [27] of the actual Internet router-level topology have reported an average degree of more than 22 per router.

For the first set of experiments, we begin with a fairly small content catalog comprising only 100 objects. The reason for this choice is as follows. When performing event-driven simulations with a large content base, the system takes much longer to come to a steady-state; while growing worse with an increasing caching budget. In such case, a large number of requests must be used just to "warm-up" the system—hence, not to be used for collecting statistics—from the initial state where all caches are empty. Besides, because of the Zipfian nature of object popularity, a larger number of requests must be generated in total to ensure that objects at the long tail of the distribution also get a reasonable chance to appear in the generated request stream. Even with a content catalog of size 100 objects with a Zipf parameter of 1, we had to generate roughly 4 million requests—while disregarding the first half—to make sure all caches in all levels have their capacity almost fully utilized before collecting statistics.

For the foregoing set-up, in Fig. 4 we compare the aggregation probability for individual objects as predicted by model versus the results from extensive event-driven simulations. Curves in each plot represent the PIT aggregation probability as attained by *each* of the content routers at the corresponding level. Thanks to the symmetry of the topology, all routers at the same level share similar statistics. Graphs in the top row contrast uniform caching against edge caching employed for graphs at the bottom. In each row, the total caching budget (CB) increases from left to right. The model accurately predicts aggregation across various caching levels even at a fine object-scale resolution. Edge caching results in higher aggregation probability at higher levels. This behavior is expected because with edge caching naturally no cache hit may occur at higher levels in the tree. Therefore, many requests that would have hit those caches if a non-zero cache size were used will now end up being aggregated at PITs.

To obtain a more insightful view of Interest aggregation, graphs in Fig. 5 show the odds of a generic Interest (irrespective of the object popularity rank) getting aggregated at each level of the tree when the link delay gradually increases. It is clear that at a fixed Interest rate, an increased link delay generally improves the aggregation probability. However, larger cache sizes tend to offset some of these improvements, especially with uniform caching strategy.

Interest aggregation occurs at a higher probability at upper levels of the tree. This can be attributed to the higher input rate into those levels considering the fact that the aggregate miss stream from many lower level routers constitutes the input of their parent router. Results from Fig. 6 suggest that significant benefits are likely to accrue from Interest aggregation; however, this promising gain should be taken with a grain of salt due to the reasons discussed in the following.

First, the small object catalog consisting of only 100 objects

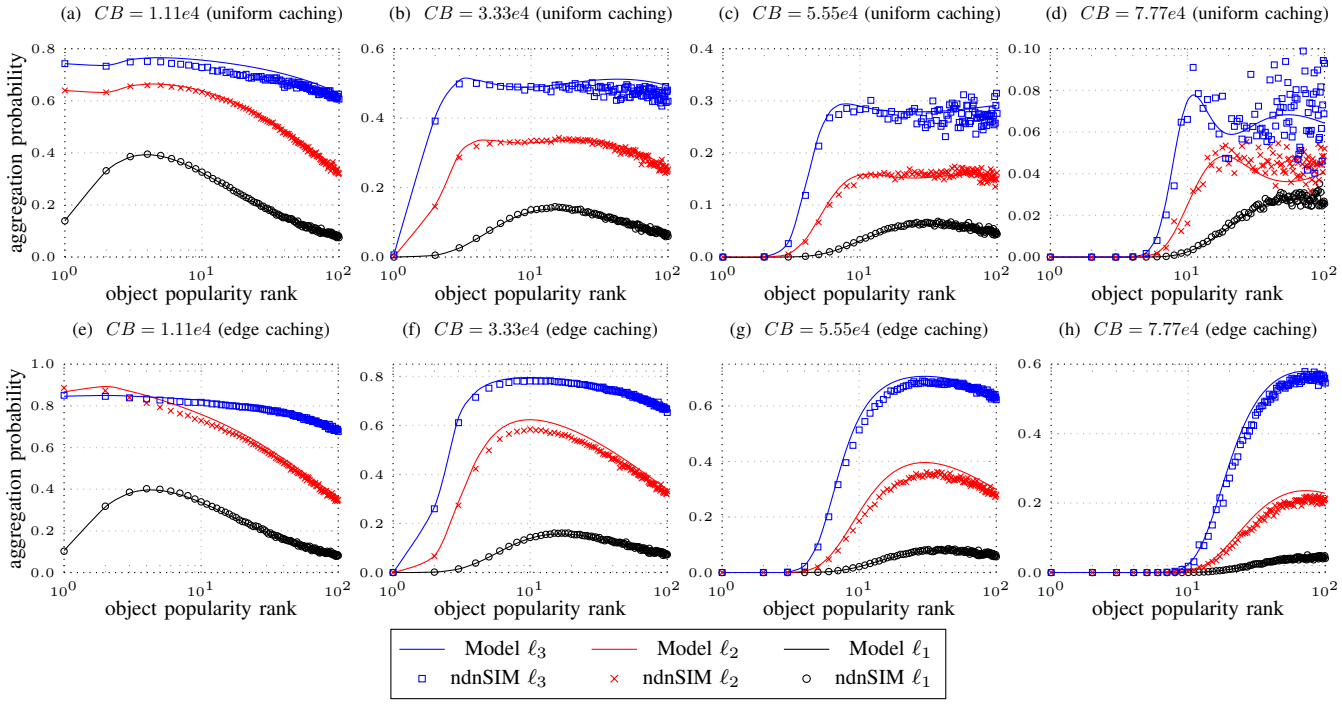


Fig. 4: A comparison of model versus event-driven simulations. Input rate into each edge router is 100 Interests/sec. Model predicts aggregation probability for individual objects fairly accurately across all levels of the tree topology.

naturally gives rise to a higher frequency of similar Interests arriving at the router, thereby an increased aggregation probability. Despite the event-driven simulation which turns out to be extremely tedious especially for a large number of objects, numerical simulations using the proposed model are practicable even on commodity hardware. Our numerical results in the next subsection confirm that the actual benefits of Interest aggregation are indeed much less in reality.

Secondly, the notion of aggregation probability itself may give a magnified image of the real benefits. In fact, aggregation probability at a certain level in the hierarchy indicates what fraction of Interests making it up to that level end up getting aggregated. Since the request stream observed by the higher level routers is a "filtered" version of the input stream to their descendants, it is clear that fewer Interests are received in total towards the top of the hierarchy. For this, we define a new measure called *aggregation percentage* that determines the percentage ratio of the count of aggregated Interests at a certain level (or at a particular router) over the total count of produced Interests in the whole system. Since every generated Interest can be aggregated at most once on its path towards the producer, aggregation percentage provides a more reasonable and unbiased measure, and we shall use that in our later assessments of aggregation benefits.

Given the foregoing remarks, we emphasize that the results demonstrated in Figs. 4 and 5 are particularly meant to verify the accuracy of the proposed analytical framework, and to provide a side-by-side comparison of how varying different parameters affects the relative odds of Interest aggregation. The true benefits of Interest aggregation are discussed in the following subsection, where more realistic input parameters are used.

B. Numerical Evaluations

Fig. 6 sheds light on the combined impact of download delay and input rate on the Interest aggregation probability. The symmetry of plots in Fig. 6 suggests that it is in fact the combination of the link delay and input rate which regulates the overall trend of aggregation probability. In fact, doubling the input rate for a fixed link delay has the same effect on the aggregation probability as keeping the input rate fixed and doubling the link delay. Therefore, we define *system load* as the product of these two quantities to build our next set of experiments on it. As a combined metric, system load does not identify a specific delay or input rate, rather defines an infinite range for these parameters. For example, a system load of 10 may imply an input rate of 100 Interests/sec with link delay of 0.1 seconds, or equivalently, an input rate of 500 Interests/sec with link delay of 0.02 seconds.

TABLE I: Table of default parameter values

Parameter	Symbol	Value
Tree height	H	5
Number of cache layers	L	3
Node degree	k	10
Total number of objects	N	140 million
Cache capacity per cache node	C	100,000 objects
Zipf exponent	α	0.8
Input rate into each edge cache	λ	100,000/sec
Link delay each way	d	15 milliseconds

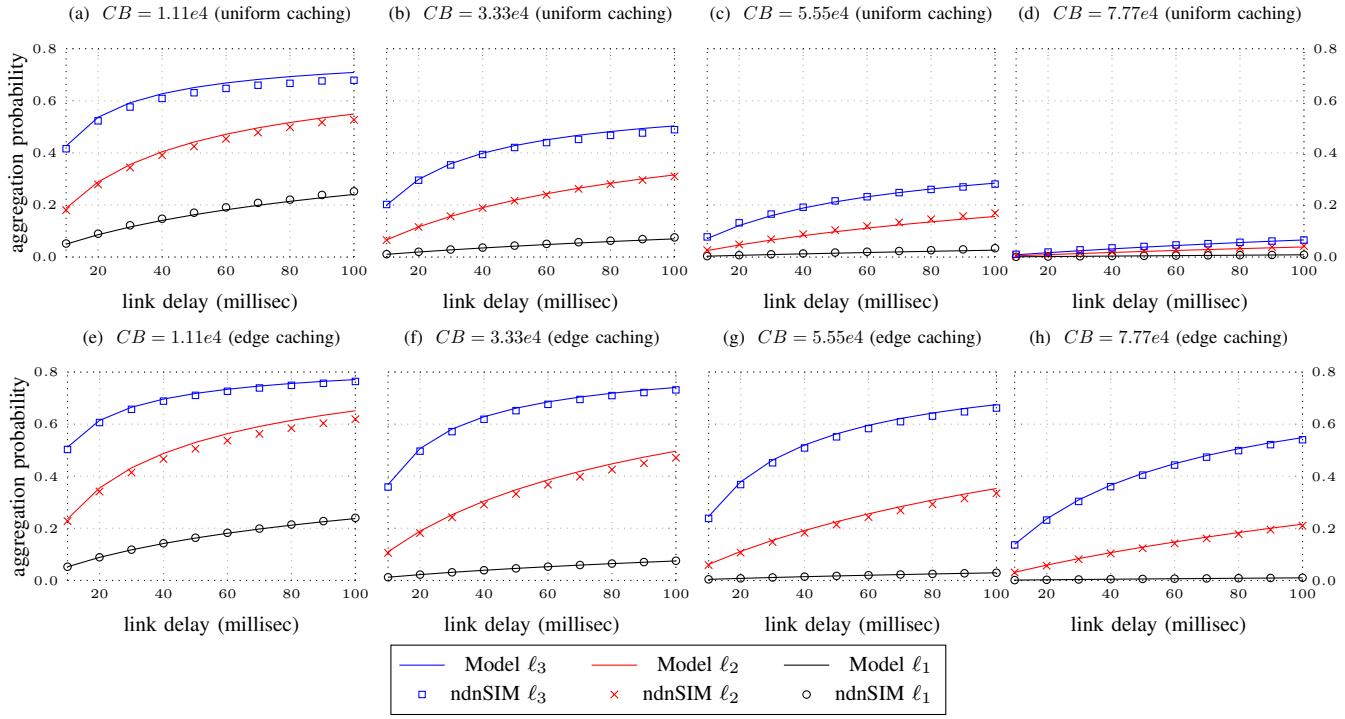


Fig. 5: Interest aggregation probability at various router levels as a function of link delay for increasing cache sizes (left to right) and different cache allocation strategies (top vs bottom rows). Input rate into each edge router is 100 Interests/sec.

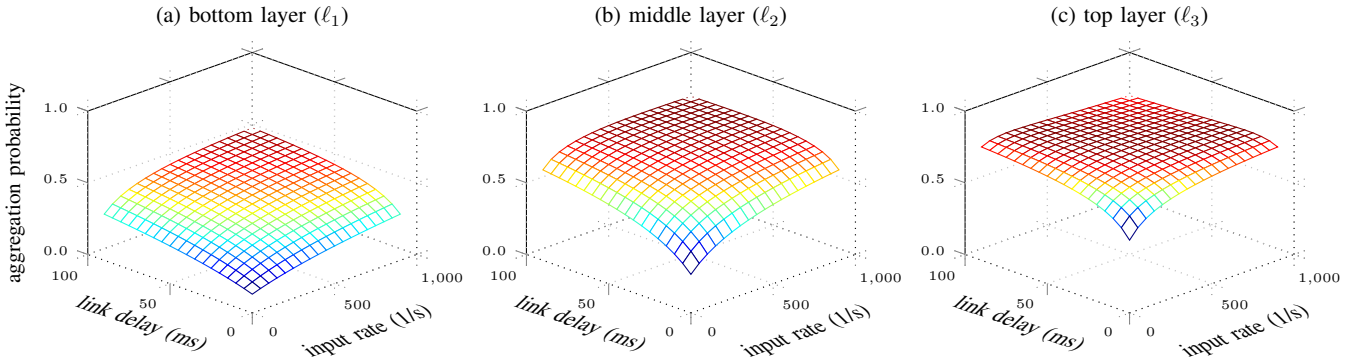


Fig. 6: The combined impact of link delay and input rate on aggregation probability. Increasing one factor has exactly the same effect on Interest aggregation as increasing the other.

In the experiments to be discussed next, we consider the same tree topology as in the previous subsection, with the general configurations listed in Table I (unless otherwise stated). The total number of objects considered, *i.e.*, 140 million, is an estimation of the total number of videos on YouTube in 2008 [28] and the Zipf parameter of 0.8 is taken from empirical studies [29], [30] of real content networks. The input rate of 100,000 Interests/sec and link delay of 15 milliseconds are also chosen such that the average generated traffic in the network is comparable with the load experienced by the Internet's backbone routers [31], [32].

Fig. 7 shows the probability of Interest aggregation at each tree level as a function of system load. Contrary to the results in Fig. 5, a side-by-side comparison of uniform- vs. edge-caching reveals that when the object catalog is large, there is no remarkable difference between these two cache allocation strategies. It is interesting that even with the highest system load of 3000, the maximum aggregation probability observed at the top most level is around 0.06. This almost 12-fold degradation compared to the previous results highlights the importance of the size of the object catalog in the overall odds of Interest aggregation. This rather surprising finding can be explained as follows. With the Zipf popularity distribution of objects, a highly popular object is requested frequently. Therefore, once such an object is downloaded into the CS, due to the frequent references to it, it stays there for a long time. Hence, Interests for that object mostly result in cache hits and are rarely aggregated. On the other hand, Interests

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

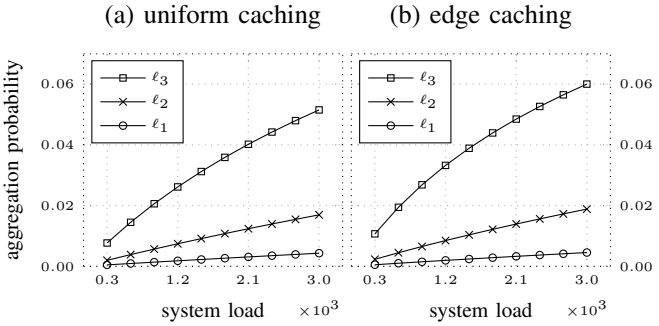


Fig. 7: Impact of system load on the aggregation probability.

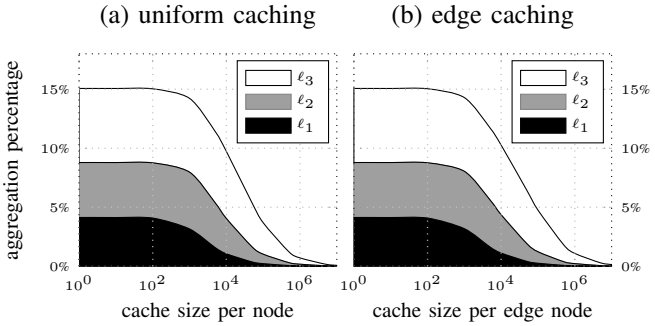


Fig. 9: Impact of cache size on overall aggregation percentage.

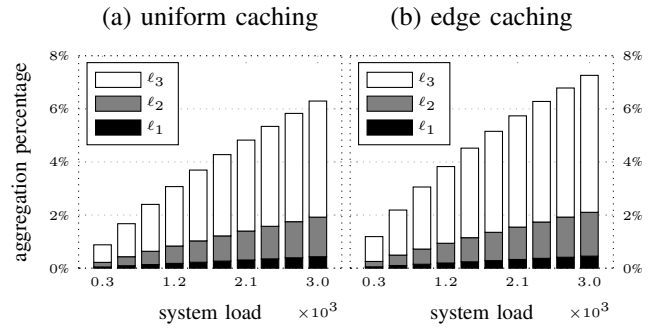


Fig. 8: Impact of system load on cumulative aggregation percentage.

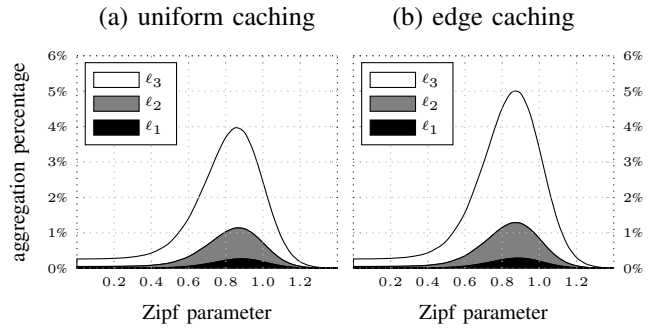


Fig. 10: Impact of popularity distribution on aggregation percentage.

for an unpopular object (in the long tail of the distribution) are received so sporadically over time that the odds of them co-occurring in the short time span when the router is awaiting the content are almost nil. As a result, in practice, Interest aggregation occurs only for a small fraction of Interests.

To make this argument even stronger, Fig. 8 shows the cumulative percentage of aggregated Interests in the system against an increasing system load. Evidently, the overall percentage of Interests being aggregated is less than 5% under a low to moderate load, and around 7% under heavy load. Note that for these results each cache node has capacity to store only 0.07% of the entire object catalog.

Increasing cache size further shrinks the benefit margin by improving the overall cache hit rates. As Fig. 9 suggests, with a small cache capacity, sizable gain (around 15% total) can be attained through Interest aggregation. However, this gain drastically decays as cache size per node increases. Eventually, with a cache size of 500,000 per node (*i.e.*, $< 0.4\%$ of the size of the content base), there is virtually no benefit in Interest aggregation. A secondary observation from the results in Fig. 9 is that with smaller cache size, all layers in the hierarchy contribute about the same in aggregation percentage; however, as more cache is added to the nodes, most Interest aggregations occur at the upper layers, while aggregation percentage at the edge approaches zero more rapidly.

Finally, Fig. 10 captures the impact of the object popularity distribution on the cumulative percentage of aggregated Interests. The non-monotonic trend of curves in Fig. 10 exhibits a diminishing returns type of effect. To explain this behavior, we note that with a Zipf popularity distribution, objects can heuristically be categorized into two groups, namely, an unpopular majority and a popular minority. The Zipf parameter (α) controls the relative size of each group as well as the skewness of the distribution. In fact, the larger the Zipf parameter, the smaller the proportion of the minority group, and the greater their popularity intensity. The latter signifies a higher access frequency to the objects in the popular group as α increases, hence a higher aggregation rate for them. However, as α increases and the proportion of the popular objects shrinks, the higher access frequency also results in higher hit rates, because a lot of those objects eventually find their way into the caches; therefore, subsequent requests for them no longer get aggregated. On the other hand, thanks to their diminishing popularity, the Interests for the majority group are also becoming so sparsely dispersed that the odds of finding a relevant entry for them at the PIT becomes negligible. For this, only a small fraction of Interests representing those fairly popular objects which may not have found a free spot in (limited-size) caches remain subject to aggregation. Further increasing the Zipf parameter shrinks down the size of the popular group gradually such that at some point, every one of them finds a permanent place in all caches. Thenceforth, the probability of aggregation becomes effectively zero.

From another viewpoint, Fig. 10 also provides suggestive evidence that even under a non-stationary content popularity distribution, no remarkable benefit can be anticipated from Interest aggregation. For example, when object references are temporally localized, a data object becomes highly popular over a certain duration of time, while its popularity gradually vanishes over time as some other data object becomes popular.

In that case, if the objects popularity is measured within smaller discrete time windows, each piece can independently be approximated with a Zipf distribution with a possibly different parameter. As Fig. 10 suggests, irrespective of how different the popularity profile looks like, only an insignificant number of Interests may be aggregated at various intervals; hence, the benefits of Interest aggregation would still remain minimal, with the maximum aggregation of less than 5% taking place around a Zipf parameter of 0.9.

V. Final Remarks and Conclusions

We presented the first analytical treatment of Interest aggregation in Content-Centric Networks using a simple yet accurate model where content download delays into the routers are non-zero. Based on our model, we introduced an iterative algorithm for analyzing a hierarchical network of content routers in terms of CS hit- and PIT aggregation probabilities and router response times. This method enables the evaluation of large-scale hierarchical caching structures, such as that of an ICN at Internet scale, with high accuracy and low computational cost for which discrete-event simulations are entirely impractical due to high processing and time demands.

Our numerical evaluations of a network of caches under realistic assumptions revealed that: (1) even with very small caching budgets, less than 5% of total Interests on average are subject to aggregation; (2) increasing caching budgets rapidly diminishes the benefits of Interest aggregation; (3) most aggregations take place closer to the producers, negating the expected benefits of reducing latency and bandwidth utilization desired from aggregation; and (4) aggregation gains are almost invariant to the choice of cache allocation strategy (*i.e.*, edge- vs. uniform-caching). Together, these observations imply that Interest aggregation should only be an optional mechanism in Content-Centric Networking. Furthermore, if per-Interest forwarding state is not needed for other purposes, the state-full forwarding plane of NDN (realized through PITs) can effectively be replaced with more efficient mechanisms, such as CCN-DART [11], [12] and CCN-GRAM [13], in which forwarding state is stored only per route or per destination while providing similar end-to-end content delivery latencies.

Our model relies on the assumption that input streams conform to the independent reference model, which need not be true in reality. However, the simulation results in [12], [13] indicate that in-network caching makes Interest aggregation unnecessary even with spatio-temporal locality of Interests.

Acknowledgments

This work was supported in part by the Jack Baskin Chair of Computer Engineering at UCSC, NSF grant CNS-1413998, and MURI ARO grant W911NF-12-10385.

References

[1] B. Ahlgren *et al.* , "A Survey of Information-Centric Networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, 2012.
[2] L. Zhang *et al.* , "Named Data Networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.
[3] [Online]. Available: http://www.ccnx.org/
[4] [Online]. Available: http://www.squid-cache.org
[5] H. Dai *et al.* , "On Pending Interest Table in Named Data Networking," in *Proc. ACM/IEEE ANCS*, Austin, USA, 2012, pp. 211–222.
[6] Y. Wang *et al.* , "Scalable Name Lookup in NDN Using Effective Name Component Encoding," in *Proc. IEEE ICDCS*, Macau, China, 2012, pp. 688–697.
[7] M. Varvello *et al.* , "On the Design and Implementation of a Wire-Speed Pending Interest Table," in *Proc. IEEE INFOCOM Workshops*, Turin, Italy, 2013, pp. 369–374.
[8] H. Yuan and P. Crowley, "Scalable Pending Interest Table Design: From Principles to Practice," in *Proc. IEEE INFOCOM*, Toronto, Canada, 2014, pp. 2049–2057.
[9] M. Virgilio *et al.* , "PIT Overload Analysis in Content Centric Networks," in *Proc. ACM SIGCOMM Workshop on ICN*, Hong Kong, China, 2013, pp. 67–72.
[10] A. Abu *et al.* , "Interest Packets Retransmission in Lossy CCN Networks and Its Impact on Network Performance," in *Proc. ACM ICN*, Paris, France, 2014, pp. 167–176.
[11] J. Garcia-Luna-Aceves, "A More Scalable Approach to Content Centric Networking," in *Proc. ICCCN*, Las Vegas, USA, 2015, pp. 1–8.
[12] J. Garcia-Luna-Aceves and M. Mirzazad-Barijough, "A Light-Weight Forwarding Plane for Content Centric Networks," in *Proc. IEEE ICNC*, Kauai, USA, 2016.
[13] ——, "Content-Centric Networking Using Anonymous Datagrams," in *Proc. IFIP Networking*, Vienna, Austria, 2016.
[14] A. Dan and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 18, no. 1, pp. 143–152, 1990.
[15] H. Che *et al.* , "Hierarchical Web Caching Systems: Modeling, Design and Experimental Results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, 2002.
[16] S. Ioannidis and P. Marbach, "On the Design of Hybrid Peer-to-Peer Systems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 1, pp. 157–168, 2008.
[17] E. Rosensweig *et al.* , "On the Steady-State of Cache Networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 863–871.
[18] C. Fricker *et al.* , "A Versatile and Accurate Approximation for LRU Cache Performance," in *Proc. ITC*, Krakow, Poland, 2012, pp. 1–8.
[19] M. Dehghan *et al.* , "On the Analysis of Caches with Pending Interest Tables," in *Proc. ACM ICN*, San Francisco, USA, 2015, pp. 69–78.
[20] J. Ni and D. Tsang, "Large-Scale Cooperative Caching and Application-Level Multicast in Multimedia Content Delivery Networks," *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 98–105, 2005.
[21] R. Chand and P. Felber, "XNET: A Reliable Content-Based Publish/Subscribe System," in *Proc. IEEE SRDS*, Florianpolis, Brazil, 2004, pp. 264–273.
[22] J. Cao *et al.* , "Internet Traffic Tends Toward Poisson and Independent as the Load Increases," in *Lect. Notes. Stat.* Springer New York, 2003, vol. 171, pp. 83–109.
[23] E. Cinlar and R. Agnew, "On the Superposition of Point Processes," *J. Roy. Stat. Soc. B Met.*, vol. 30, no. 3, pp. 576–581, 1968.
[24] A. Conn *et al.* , *Trust Region Methods*, ser. MPS/SIAM S. Optimizat. Philadelphia, USA: SIAM, 2000.
[25] S. Gratton *et al.* , "Recursive Trust-Region Methods for Multiscale Nonlinear Optimization," *SIAM J. Optimiz.*, vol. 19, no. 1, pp. 414–444, 2008.
[26] A. Afanasyev *et al.* , "ndnSIM: NDN Simulator for NS-3," NDN, Technical Report NDN-0005, 2012. [Online]. Available: http://named-data.net/techreports.html
[27] "Internet Topology at Router- and AS-levels," accessed: 2015-07-17. [Online]. Available: http://www.caida.org/research/topology/generator
[28] A. Russakovskii, "How to Find Out the Number of Videos on Youtube," 2008, accessed: 2015-07-22. [Online]. Available: http://beerpla.net/2008/08/14/how-to-find-out-the-number-of-videos-on-youtube/
[29] A. Mahanti *et al.* , "Traffic Analysis of a Web Proxy Caching Hierarchy," *IEEE Netw.*, vol. 14, no. 3, pp. 16–23, 2000.
[30] C. Fricker *et al.* , "Impact of Traffic Mix on Caching Performance in a Content-Centric Network," in *Proc. IEEE INFOCOM Workshops*, Orlando, USA, 2012, pp. 310–315.
[31] J. Cowie *et al.* , "Modeling the Global Internet," *Comput. Sci. Eng.*, vol. 1, no. 1, pp. 42–50, 1999.
[32] C. Fraleigh, "Provisioning Internet Backbone Networks to Support Latency Sensitive Applications," Ph.D. dissertation, Stanford University, Stanford, USA, 2002.

Let's Collect Names: How PANINI Limits FIB Tables in Name Based Routing

Thomas C. Schmidt, Sebastian Wölke, Nora Berg
Dept. Informatik, HAW Hamburg
firstname.lastname@haw-hamburg.de

Matthias Wählisch
Freie Universität Berlin
m.waehlisch@fu-berlin.de

***Abstract*—Name-based routing as proposed in Information Centric Networking encounters the problems of (a) exploding routing tables, as the number of names largely exceeds common routing resources, and (b) limited aggregation potentials, as names are commonly independent of content locations. In this paper, we introduce Partial Adaptive Name Information in ICN (PANINI), an approach to scale routing on names. PANINI aggregates names at (virtual) collectors and adapts FIB tables simultaneously to available resources and actual traffic patterns. PANINI introduces routing hierarchies and prefix-specific default routes, bimodal FIBs, and confined flooding. We thoroughly evaluate the approach in theory and practical experiments. Our findings indicate that effective reductions in control state largely outweigh overheads in control traffic.**

***Index Terms*—FIB aggregation, scalable adaptive forwarding, CCN/ NDN, confined Interest flooding.**

I. Introduction

Information Centric Networking has introduced a new, promising communication paradigm, but continues to struggle with severe challenges [1]. NDN [2] (among others) binds routing on names at a high level of maturity. However, the multitude and complexity of distributed content names has not been treated convincingly. Names are by orders of magnitude too many to be stored in today's forwarding information basis (FIBs) and remain too delocalized to allow for aggregation. Even though several original approaches have been presented [3], [4], the sheer scalability demands risen from names prevent a striking step forward.

Scalable routing in the current Internet is achieved by a hierarchy that shields global from local operations. The majority of route identifiers is situated at the edge, but treated as aggregates at the core. The initial concepts of routing on hierarchical names invert this principle and require detailed, de-aggregated knowledge of name state throughout the network. In this paper, we approach this problem by introducing Partial Adaptive Name Information in ICN (PANINI).

The PANINI approach [5] starts from fixing an aggregation point for a group of names resident in a (topological) network. The typical aggregator would be a larger cache repository on the fixed Internet, or a gateway in the IoT. We assume topology building mechanisms in place that generate a shortest path tree rooted at the aggregation point. This is in full analogy to the current Internet, where standard routing protocols can construct shortest paths on the inter- and intra-domain level. Given this basic topology, every node can identify up- and downward paths with respect to the aggregating root—with upward paths serving as default.

The objective of name-based routing is to link content requesters with content suppliers in an efficient way. Inspired by the highly skewed popularity distribution of names, PANINI aspires to efficiently balance FIB sizes and control traffic. Popular names are included in distributed tables, while unpopular ones are omitted and searched by confined flooding. Our thorough evaluations reveal significant optimizations at small FIB tables *and* rare flooding events.

In the following, we will present this hybrid combination of (artificially enhanced) name aggregation at rendezvous points, adaptive mapping by FIBs, and a dynamic on-demand flooding of Interests towards content suppliers. We start with a problem statement and discuss related work in Section II. The PANINI routing and forwarding scheme is presented in Section III along with several deployment scenarios. Extensive evaluations accounting for both, theory and experimentation follows in Section IV. Finally, we conclude and give an outlook in Section V.

II. The Problem of Scalable Routing on Names and Related Work

A. FIB-size, Aggregation, Flooding

Scalable name-based routing is one of the open research challenges in ICN [1], [4], [6], [7]. This problem appears at least with two faces—limiting state (FIBs sizes) and control traffic at routers.

A common approach to reducing routing entries is aggregation. Aggregation of names, though, requires a correspondence of identifiers and locations. Such an assumption conflicts with a flexible or distributed content placement within the network. According to current common practice, content names belong to the content owner and not to the network provider. Consequently, a content owner can decide to change the ICN upstream, which then leads to de-aggregated routing entries. Furthermore, routers on names in ICN cannot locally decide on aggregation, since names—

ISBN 978-3-901882-83-8 © 2016 IFIP

unlike IP addresses—do not fall into a fixed, enumerable number space.

The Internet consists (and will consist) of heterogeneous kinds of routers in terms of hardware and capacity. Assuming a flexible name-based content distribution system—as originally envisioned in the ICN community—routing tables will naturally aggregate rather sparsely and easily cause memory exhaustions at most routers. As long as a router is single-homed, all entries can be collapsed to a (default) entry towards the (single) uplink. Most mobile end devices as well as edge routers are multi-homed, though. In PANINI, we extend the concept of default routing and leverage its benefits without ignoring the potential of the underlying network structure.

An alternative approach to implement scalable routing is to separate names from locators and deploy a name resolution service [8], [9]. Staying with the core concepts of NDN/CCN and its security benefits, PANINI performs routing solely on names and without a mapping.

Converse to a complete table view or a default routing system operates a path detection by flooding. Flooding helps to explore the location of content but is clearly not applicable on Internet-scale. PANINI exploits flooding occasionally in strictly confined local regimes, when hardware resources are limited.

To reduce the amount of memory allocated by FIB entries, several data structures have been proposed that are specifically tailored to name-based routing (e.g., [10]). We consider those approaches orthogonal to PANINI, as they help to implement scalable name-based routing (wherever a complete view is required) but do not solve the scalability problem from first principle.

B. Name-based Routing

Recently, the debate on how to improve the state of name-based routing has heated again with several proposals. OCEAN [11] starts from the observation that aggregation is unlikely to occur on its own at Internet scale. Agreeing with this observation, we introduce aggregation facilitators. Instead, OCEAN proposes to aggregate on virtual paths and (re-)introduces a virtual circuit path switching facility. These 'pathlet'-type forwarding also eliminates loops that can occur in current NDN/CCN. By introducing a clear, Internet-type route hierarchy, loops are equally prevented in PANINI.

SNAMP [12] proposes a *map-encap* approach including mapping services at the edges, which link an arbitrary name to a backbone-specific prefix. Thus, routers in the default-free zone (DFZ) need to store only a subset of prefixes of the overall namespace but the edges need to handle full tables. SNAMP introduces inverse requirements as compared to PANINI where name collectors with complete FIB entries span the DFZ.

Wang *et al.* [13] designed a flooding mechanism for ICNs. Therein the flooding radius is set dynamically from local graph properties that yield information about the global structure. It shows that in scale free networks nearly all information can be retrieved within a very small flooding radius. In PANINI, we apply additional restrictions and can show that even in the worst case, the overhead which is introduced by flooding remains relatively moderate.

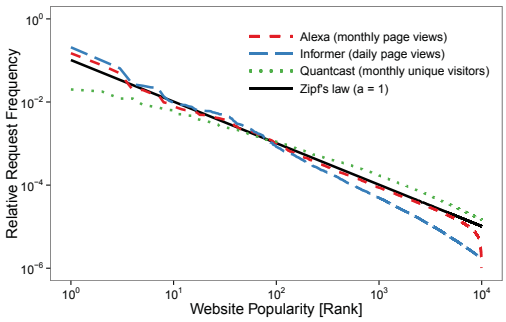


Figure 1. Zipf's law in comparison to empirical name popularity from different sources

Geometric routing [14]–[16] addresses the problem of route scalability by encoding forwarding in coordinates. These approaches are promising as they do not require a global routing table, neither in the edge nor in the core. However, embedding arbitrary names in geometric space is still an unsolved problem in real Internet-like deployment.

C. Name Popularity

The huge numbers of names for content can be contrasted by its largely uneven frequency of use, which PANINI exploits. The distribution of name popularity has been repeatedly measured in different contexts like web caching [17], or web access [18] and was found to be a power law distribution of Zipf type [19].

For confirmation and parameter fixing, we performed additional checks on web data of different type and periods. In detail, we consulted the three different web analytic services *Alexa*[1], which provides monthly page view statistics for the top million websites, *Quantcast*[2], which offers statistics of unique monthly website visits from the United States, and *Informer*[3], from which we crawled the daily visitors and page views per website.

Results are displayed in Figure 1 in comparison with Zipf's law for $a = 1$. All measurements remain in reasonable agreement with the Zipf curve, why we continue to build our content popularity model and analysis on it.

D. Modeling Shortest Path Trees

The routing mechanism of PANINI creates shortest path trees (SPTs). From theory [20], [21] we know that SPTs are well modelled by uniform recursive trees (URTs). URTs are random trees that can be generated stepwise as follows. First, the root vertex is added to an empty graph. In each

[1]http://www.alexa.com
[2]http://www.quantcast.com
[3]http://website.informer.com

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

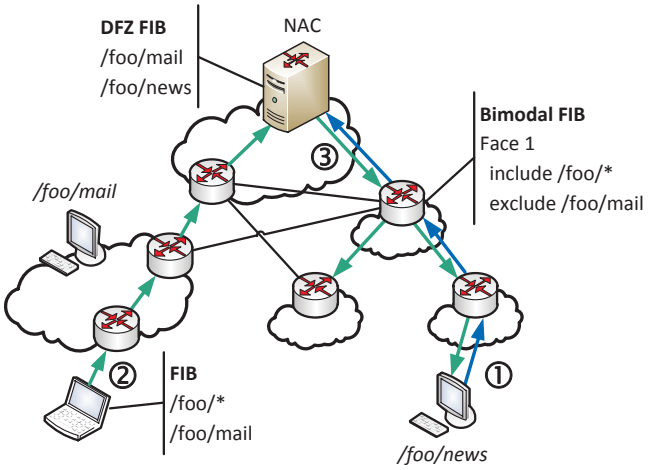


Figure 2. Overview of PANINI routing and forwarding.

following step n, the n-th vertex is connected to one of the $n-1$ previous vertices with equal probability [22]. The resulting tree is self-similar and represents the structural properties of shortest path trees within networks, as we will exploit in our analytical evaluation.

III. PANINI Routing

A. Overview

The core ideas of PANINI are to limit global as well as local FIBs by introducing aggregation points (*name collectors*) that are reachable via default routes, and an utility-adaptive FIB management.

1) Limiting Global FIBs: PANINI distributes responsibility for names across Name Collectors (NACs) by assigning prefixes to NACs. Each NAC aggregates those names that match its prefix(es) at a static preconfigured position in the network. Prefixes can be selected demand-wise and distributed among an arbitrary number of NACs, tailoring a partition of the routable namespace according to content popularity, topological preferences, or provider needs. We discuss diverse deployment scenarios in the following subsection.

As such, a NAC serves as a prefix-specific anchor point that aggregates name-based routes and facilitates name-specific caching. The anchor itself becomes reachable via prefix-specific default routes, which a NAC simply advertises within its domain. These default routes purely depend on the topology and remain independent of individual content providers. Following such a default route for a prefix, any node within the network can reach the corresponding NAC on the shortest available path. Hence the union of all default routes for a prefix will define a shortest path tree rooted at the NAC. Figure 2 displays such a default for the prefix `/foo/*`. From this perspective, NACs form the default-free zone (DFZ) in PANINI routing.

2) Publishing Content: A content supplier who wants to advertise a name to the routing system uses the default route towards its most specific prefix for issuing a Name Advertisement Message (NAM) (see step 1 in Fig. 2). Per default, NAMs travel hop-by-hop towards the aggregation point, and every intermediate router can harvest the content advertisement for including in its own routing table. These table entries are specific, down-tree oriented non-default routes. Filling all FIBs will generate a complete routing path from the aggregation point to the content source. It is worth noting that routing states close to a NAC naturally aggregate in FIBs.

3) Requesting Content: A consumer requests content by transmitting an Interest for a name. In the absence of more specific FIB entries or cache hits, this Interest will travel up to the NAC on the default route (see step 2 in Fig. 2), where popular content is likely to be cached. If not satisfied from the cache, the Interest will be forwarded down along the previously installed path to the content provider (see step 3 in Fig. 2). Data forwarding will follow the pending Interest states on the reverse path as in regular NDN/CCN. Routing and forwarding are thus aligned to a network hierarchy that resembles the current Internet with aggregation points located at the transit tier. It is noteworthy that routing towards the NAC will aggregate paths and thereby facilitate on-path caching.

4) Limiting Local FIBs: Up to this point, we have globally reduced FIB entries to prefix-specific defaults, but required names present in local FIBs. This is known to be infeasible in ICN. PANINI weakens this requirement as follows. Complete routing tables shall only be required at the aggregation points. This is a significant relaxation, since aggregation points are designed to facilitate name aggregation and largely reduce routing table space. In addition, providers may select strong NAC devices. From complete, aggregated FIB tables, the (transit) root can thus always tell which branch (or lower tier ISP) holds the requested content. Without further FIB entries, flooding may lead the Interest down this (loop-free) branch.

Intermediate nodes are not required to carry a full FIB, but rather aim at adapting selected entries to minimize Interest flooding. In analogy to caching content, each node autonomously decides about (a) its memory resources available for the FIB, and (b) the forwarding logic it applies within its vicinity. Traffic flows (with highly skewed name utilization) can be continuously used to adapt the FIB to relevant traffic patterns. For example, a node can hold more specific information for frequently requested names, while it may erase entries for traffic rarely seen.

To optimize Interest guidance with partial forwarding information even further, we introduce a *bimodal FIB*. This extends the FIB structure to operate in two modes—`include` and `exclude`. In `include` mode, all Interests that match a FIB prefix will be forwarded on the associate Face, while all Interests that match a FIB `exclude`-prefix will be blocked on that Face. The initial state of an empty FIB reads `exclude *` which prevents flooding of all incoming Interests. A node that has seen no routable names from

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

NAMs about a prefix `/foo/*'` in a subtree of his will remain `exclude /foo/*`.

In combining these two routing mechanisms—(i) default prefix routes to NACs and (ii) adaptive bimodal FIB management—the PANINI system largely reduces FIB tables. In the evaluation Section IV, we will show that even with constant, small FIB sizes the routing remains highly efficient.

B. Deployment Scenarios

1) A Content Delivery Internet: In an Internet-wide deployment, PANINI content distribution will rely on per prefix replicated NACs that are placed in various provider networks. Each NAC serves its local content suppliers and requesters as the default addressee. Routing and forwarding for locally available or cached content remains local as described in the previous section.

To make content accessible across domains, NACs in service of the same prefixes need to peer in the default free zone. NACs need to exchange more specific prefixes of their local content names, which can be done similar to BGP network prefix exchange, or by a distributed key-value system. It is worth recalling that NACs are aware of their complete local name tables and can therefore efficiently aggregate. Request routing and content forwarding can then be performed either by directly traversing the local NAC and its peerings, or recursively by the local NAC as it is common in today's CDNs. A detailed study of routing and forwarding in the DFZ will be subject to our future work.

2) ICN in the Internet of Things: IoT networks commonly consist of distributed sensors and actuators which are often constrained, embedded devices, and at least one (full-featured) Internet gateway. To serve this setting, we first need to create a topology. We propose to follow the well-established approach of building a tree-like structure—a destination-oriented directed acyclic graph (DODAG)—as known from RPL [23], for example. Parents broadcast their presence (DIO) and children attach (DAO). These link-local operations can be transfered to the link-layer in a straight-forward manner. The IoT gateway takes the role of the root node and NAC.

Given this basic topology, the gateway(s) can announce their default prefixes, which in a simple network will reduce to a unique default route (`/*`). In the IoT, we need to face the trade-off that Interests in a constrained environment should ideally be minimized, but intermediate nodes have limited memory and cannot hold large routing tables. In our previous work [24], we have designed and analysed two routing corner stones—Vanilla Interest Flooding (VIF) and Reactive Optimistic Name-based Routing (RONR). While VIF works without a FIB, RONR nodes gradually acquire all FIB entries in a reactive fashion. Given the DODAG topology, PANINI can now define a self-optimizing strategy for routing on names by making a hybrid use of both routing primitives—typically keeping the FIB limited to hold a few entries that are replaced according to a least frequently used (LFU) policy.

Table I
EXAMPLE OF A PANINI FIB TABLE

Mode	Prefix	Face
`Default`	`/foo/*`	`1`
	`/bar/*`	`1`
`Include`	`/my/videos/`	`2`
	`/your/music/`	`3`
`Exclude`	`/qux/*`	`2,3`

3) Edge Caching in 5G Mobile Networks: The emerging 5G mobile network architecture foresees an ultra dense access network that is backed by a shared data domain for fast content access. Several major players including Cisco opt for deploying ICN technologies to facilitate content caching at the edge.

PANINI routing will significantly simplify deployment as follows. NACs shall be positioned as virtualized networked functions in the shared data domain to channel content retrieval and caching from the open Internet (either by a name-based peering or by traditional IP). NACs can dynamically resize in their virtualized environment to adapt caching capabilities to content popularity. They will distribute default content routes throughout the access network so that the ultra dense access only needs to carry a minimized set of FIB entries that persist with topology. Neither complex, user-driven route management nor flooding are required, as content is always pulled from the data sharing domain.

This setting resembles the base PMIPv6 multicast architecture [25], which experienced deployment. Here access routers (MAGs) play the role of request proxies and regional mobility anchors (LMAs) serve as content aggregation points.

C. Initializing a Default Distribution System

To illustrate the PANINI routing system in detail, let us consider an initial network prior to any signaling. This system consists of interconnected routers and a collection of NACs. NACs have prefixes assigned and routers preconfigured their FIBs autonomously to `exclude /* *` (cf., Table I). The initialization of the distribution system then proceeds in three phases.

1) Setting-up Defaults: Once configured, the NACs will start to announce their prefix availability and thereby construct shortest path trees (SPTs) per prefix. Efficient protocol mechanisms for that task are well known such as BIDIR PIM [26] in the Internet, or RPL [23] in the IoT. These SPTs are defined by the corresponding default entries in regular FIBs (cf., Table I). It is noteworthy that defaults enable any router on path to distinguish upstream and downstream messages prefix-wise.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

2) Registering Content Names: On completion of Phase 1), content is only available at or via the NACs (e.g., from external peering). In order to publish content, providers need to register their content names with the appropriate NAC. For this providers issue a Name Advertisement Message (NAM) that is forwarded along the default route to its NAC. At each hop, intermediate routers see such advertisement and must update their FIBs to maintain consistency. For example, an 'exclude all' interface must change state to include the newly seen name or prefix. Alternatively, forwarding prefixes that have been already configured may need extension to include the new advertisement. Beyond consistency demands, an intermediate router is free to decide about the level of precision it includes in its Forwarding Information Base. To ensure message redundancy, resilience, and to facilitate adaptive FIB management, NAMs need periodic retransmission as is common in related Internet protocols.

3) Processing Content Interest: In the third phase, all content is available for request as described in Section III-A. Interests arriving at a router face are placed in the Pending Interest Table (PIT) like in regular NDN/CCN, and are forwarded under aggregation according to the FIB. However, as FIBs need not be complete, Interest forwarding needs to adapt in the following way.

Any Interest traveling up-tree will be forwarded along default routes, unlike a *specific* FIB entry (i.e., 'include' as in Table I) refers downwards. It should be noted here that longest prefix match cannot be applied to name-based routing without globally coordinated aggregation. At the NAC, a complete FIB will guide the Interest down to its dedicated subtree. Arriving on the downward path, any intermediate router will search its FIB for a specific route. If present, the Interest will travel in regular unicast mode. In case of a FIB miss, the router will select all down-tree interfaces without a matching exclude entry for broadcasting the Interest. We will show in Section IV that the expected broadcast fanout is small, and—by the recursive nature of SPTs—independent of network size. Note that loops are strictly prevented as Interests travel up-tree only once and monotonically downwards thereafter.

Additionally, Interest arrival is a measure of content popularity and used to adapt the FIB population at intermediate routers. We will discuss such adaptive FIB management in the subsequent section.

D. Adaptive FIB Management

A major objective of PANINI is to effectively limit FIB sizes. This is at first achieved by enhanced aggregation and default routes, but strict limits require additional measures. We now address how local routers can independently limit their FIBs in an optimized fashion.

A PANINI router can impose strict limits on its FIB, the minimum being a single default `/*`. While admitting incomplete forwarding information by flooding, it is the idea to keep broadcasts unlikely by populating the FIB according to content popularity, which is highly skewed (see Section II). PANINI does not dictate a common policy for managing FIBs, but rather leaves this to individual capacities and configurations of nodes. At the same time, any on-path router can measure name popularity through Interest processing and maximize the utility of its table entries.

We favour two strategies for an adaptive FIB management, leaving the field open, though, for further strategic improvements. A minimal approach—also feasible in the IoT—is to fix a table size and replace by least frequent use. In detail, FIB entries keep a statistic counter that is incremented for every Interest match. When a new name advertisement arrives, it replaces the least popular FIB entry, leaving the total size unchanged. Additional thresholds in frequency and time can increase convergence of this simple scheme.

A relaxed scheme based on soft states may be preferable at moderately constrained FIB sizes. Every advertised name will at first written to the FIB with a timestamp attached. A FIB entry then will expire after a timeout period, unless an Interest refreshes its timestamp. In this way, FIB tables will adjust to content variety and request frequency, possibly fluctuating heavily in size. The actual FIB properties may be fine-tuned by adjusting the timeout period or imposing additional thresholds.

IV. Evaluation

In this section, we evaluate PANINI with a special focus on routing costs and overheads. We will approach the subject from two sides, theoretically by analysing the structural properties of the routing trees, and experimentally by emulating virtual PANINI networks in our lab. While theory grants rigorous insights into intrinsic characteristics of the system and to scale its size, experiments practically reveal net effects from the different constellations of the huge state space. Wherever possible, we compare results.

A. Theoretical Modeling

The PANINI routing scheme is built on prefix-specific shortest path trees (SPTs) that are rooted at the corresponding NACs. Without loss of generality it suffices to analyse the properties of a single tree, as no further assumptions are made w.r.t. individual prefixes.

1) Flooding Fanout: A router on the shortest path may experience a FIB miss in PANINI and needs to broadcast an Interest. In the absence of `exclude` entries, the flooding overhead increases with the fanout. Fanout resp. degree distributions are known for URTs [22]. Consider a URT of N nodes, then the expected number of nodes with degree k can be approximated by

$$E\left[V_N(k)\right] = \frac{N}{2^{k+1}} + \mathcal{O}((log\ N)^{k+1}). \qquad (1)$$

Figure 3 shows the normalized degree distribution for different network sizes which coincide due to the recursive

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

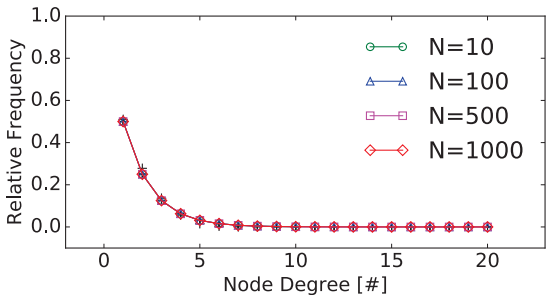


Figure 3. Node degree distribution for URTs of different sizes

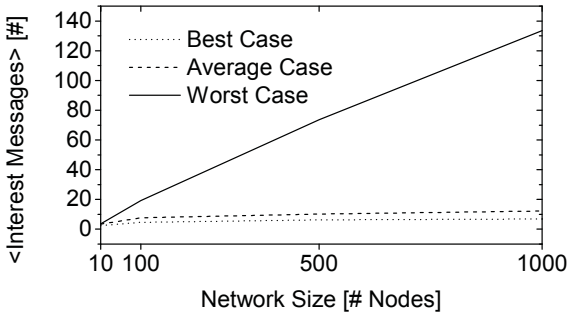


Figure 4. Average number of Interest messages for different scenarios and network sizes

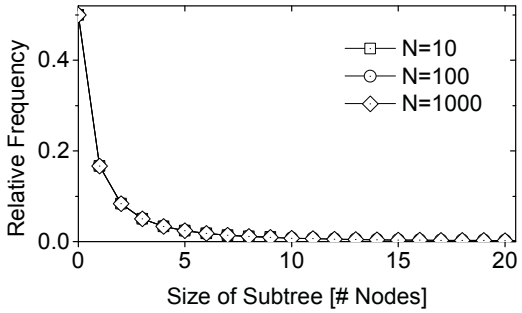


Figure 5. Distributions of branch sizes in a routing tree of N nodes

nature of the trees. In particular, fanouts (= degree - 1 uplink) are uniform and likely range between one and five. This confirms a limited ramification of URTs that are uniformly wide but short, and a broadcast at a single node will have limited multiplicity.

2) Flooding in Subbranches: When a PANINI node floods downwards after a FIB miss, there are two options of Interest propagation at each receiver. Either it holds a matching FIB entry, or it continues flooding. By the latter, flooding may extend over complete subbranches, which again share the URT property at reduced size. In this section, we calculate the worst, best, and average number of nodes that receive a flooded Interest message.

In the worst case, a direct child of the NAC holds no forwarding state for a specific Interest message and initiates flooding. In the absence of any further forwarding information, the nodes which receive the broadcast message are all in that branch. The expected size of such a branch can be calculated from the number of nodes and the expected root degree as follows.

Let U_N denote the random variable that maps from an URT with N nodes to the degree of the root vertex. Then the expected root degree reads [27]

$$E[U_N] = \sum_{j=1}^{N-1} \frac{1}{j} = H_{N-1} \tag{2}$$

that is the $(N-1)^{th}$ harmonic number H_{N-1}. Since the distribution of branch sizes is uniform in an URT, we obtain the expected branch size from dividing the remaining $N-1$ by (2).

A Uniform Recursive Tree of N nodes has an average depth of $\log(N)$, which is the optimal number of downtree Interest messages. For the average scenario, we consider the unicast hops and a random FIB on path empty with its corresponding subtree flooded (see below). We visualize the outcome in Figure 4. As can be seen from the graph, the average number of Interest messages needed for locating content follows closely the logarithmic behavior of the best case. In contrast, the worst case scenario grows almost linearly ($\approx N/\log N$).

We now calculate the size distribution of a randomly selected subtree following the Polya urn model. We consider N nodes that are numbered in the order of attachment. Let $D_N(k)$ denote the number of descendants of node $k > 1$, then the distribution can be derived from the Polya-Eggenberger distribution [28], [29]

$$\mathcal{P}(D_N(k) = j) = \frac{(k-1)\cdot(N-k-j+1)^{\overline{j}}}{(N-j-1)\cdot(N-1)^{\overline{j}}}, \tag{3}$$

with $(n)^{\overline{j}}$ the j-th rising factorial power of n.
Summing over all nodes k with equal probability $1/N$ yields the distribution D_N of nodes in a subtree rooted at an arbitrarily chosen node.

Figure 5 visualizes these analytical distributions for different numbers of nodes. Strikingly, the branch sizes are largely independent of the overall network size, which is due to the recursive nature of the URTs. Node numbers from these exponentially decaying distributions are rather small: more than 7 nodes appear with probability 0.01. This is again due to a uniformly wide fanout—trees are rather wide and short.

3) Resilience and Robustness: To provide robust data distribution a network needs to adjust to changes and failures. PANINI operates on the basis of shortest path trees and needs to cope with nodes that disappear from the SPT and possibly reappear somewhere else on the tree at a later time. Therefore, the FIBs need to be updated, e.g., by periodically repeating publishing messages. With the insertion depth of a node in an URT, we can estimate how many link changes are required and thereby how large

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

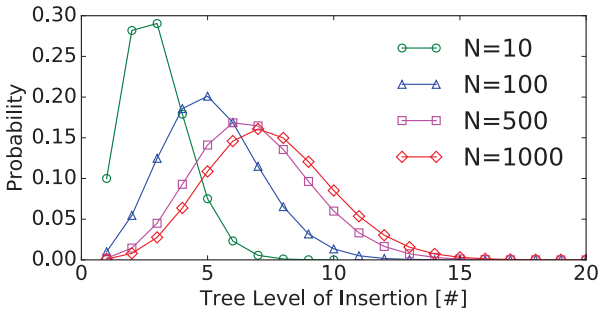
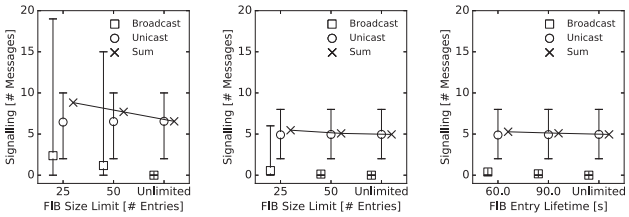


Figure 6. PDF of the insertion depth of a failing node



(a) URT (b) AS 1239 (sized) (c) AS 1239 (timed)

Figure 7. Overall distribution of message types for different FIB sizes resp. FIB entry life times, showing average, 95% and 5% percentile

a damage and the effort to update the FIB at the NAC are.

To obtain the insertion depth of a node, we use the profile $E[V_{d,N}]$ of a URTs [22], which describes the expected number of nodes at level d. Let random variable I_N map to the insertion depth d of a node, then

$$P(I_N = d) = \frac{E[V_{d-1,N-1}]}{N-1} \tag{4}$$

Figure 6 depicts the probability density functions of I_N for different network sizes. The graph shows that the insertion depth of a node only slowly increases with growing size. The reason is again that the shapes of URTs tend to grow relatively wide but not very deep. Furthermore, in URTs about half of the nodes will be situated at level $\ln(n)$. That means that we expect a new publish message to travel $\ln(n)$ hops until the NAC has the right information in which subtree the node rejoined.

B. Experimental Emulation

We now proceed to our experimental evaluations that are performed using a realistic emulation environment. While the theoretical analysis remained limited to structural properties of the routing system, we experimentally consider real-world topologies, realistic name popularities, and adaptive FIB management. Caching was not considered in this work, since arbitrary cache effects would blur the outcome of routing and FIB management that we want to reveal. However, adding caching to the system will only improve the overall performance.

Evaluations again concentrate on the routing costs and overheads produced by PANINI. In addition to control messages and flooding costs, we quantify the path stretch for the Rocketfuel topologies, which also serves as an indicator for overheads in forwarding delay.

1) Experimental Environment: For experimentation, emulated networks were set up on a 64-core host machine based on virtual nodes and Mininet [30]. Topologies were created from Rocketfuel data [31] and from artificially generated URTs.

Every virtual node ran an NDN-Forwarder based on a modified version of NDN (0.4.0-beta2)[4]. PANINI modifications were implemented in the NDN strategy layer which allows for specific processing of each individual packet. We implemented a fixed size FIB operating a LFU replacement policy and a life-time management for FIB entries. Using this setting, we were able to reliably run networks of several hundred core routers without loosing packets or exhausting resources otherwise.

All individual experiments were performed according to the following scheme. We fixed a topology and FIB size, and placed the NAC at its center (on the node of highest betweeness). For each content name from a set of 10.000, we placed providers on random but fixed routers in the topology. Consumers were emulated as child nodes of the core routers. Each consumer was placed uniformly random and requested content from our name set according to a Zipf distribution. One million individual message paths were iterated, monitored, and recorded for evaluation.

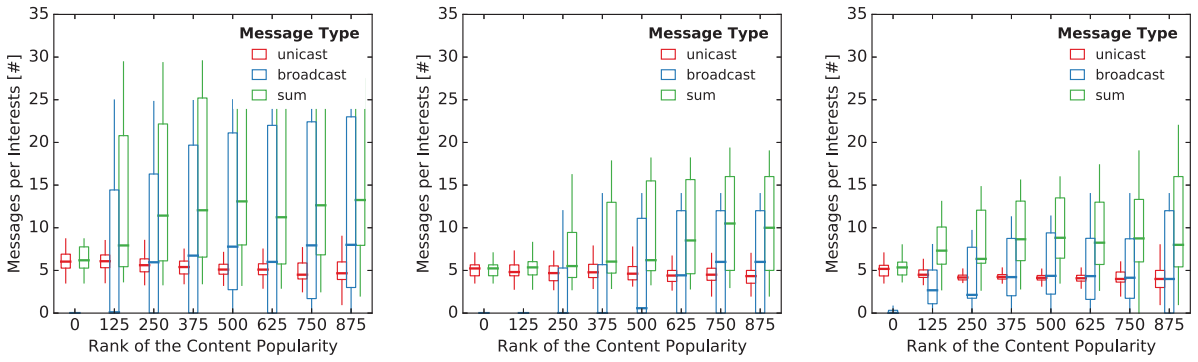
C. Experimental Results

1) Expected Message Distribution: Our first glance at the system addresses the overall messaging behaviour. We are interested in the average number of unicasts and broadcasts per content request for different topologies and FIB sizes, as well as FIB entry life times. The results in Fig. 7 surprise with a remarkably low broadcast appearance for both, the artificial URT (size 100 routers) and the AS1239 Rocketfuel topology (size 315 routers). Only for small FIB sizes in the URT, broadcast multiplicities fluctuate by an order of magnitude. With increasing FIB sizes, but in particular for the temporal FIB entry management, flooding reduces to about a single broadcast per request. Note that the routing refresh rate equals 200s and the average path lengths in both topologies is close to six.

For a more differentiated view, we correlated the message type distribution with content ranks (see Fig. 8). Unsurprisingly, broadcast multiplicities and fluctuations largely increase with decreasing popularity. In detail, content requests above rank 100 caused visibly fluctuating broadcasts. However, given the heavily skewed content distribution

[4]http://named-data.net/

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



(a) URT (b) AS 1239, FIB size limit 50 (c) AS 1239, FIB Entry Lifetime 60*s*

Figure 8. Measurements of (typed) message frequencies for selected popularity ranks and different scenarios

known from Fig. 1, it becomes evident that these flooding events carry limited weight and contribute little to the overall average results in Fig. 7.

These numbers strongly support the initial PANINI assumption that low-ranked content names in FIBs are of little use.

2) Flooding in Subtrees: The following analysis explores the empirical shapes of flooded subtrees, which had already been discussed in the theory section. For the same experiments, we identify (a) all coherent subtrees and (b) the accumulated size of flooded regions that are composites of a larger tree with subtrees connected via unicast links. The latter are results from alternating FIB misses and hits along paths.

Fig. 9 visualizes the different distributions of subtree sizes. Results for single broadcast trees only, are in excellent agreement with theory and very small. Almost 90 % of flooded subbranches subsume less than 10 nodes. This again reflects the limiting characteristic of the recursive structures. On the contrary, composite trees tend to be much larger with about 50 % exceeding 25 network nodes. This is an indication of fluctuating decision at neighboring routers that cannot converge on treating certain names. Even though these events occur rarely and carry little weight, we expect to improve this behavior with name aggregation at FIBs close to the NAC. Name aggregation has not been implemented yet, but shall assign an increasing weight of names at up-tree routers.

3) Path Stretch: In most cases, PANINI transmits Interests via a NACK to the producer (in the absence of caches) and thus may artificially extend paths. To quantify this effect, we evaluate the distributions of path stretches for all Rocketfuel topologies. The results are displayed in Fig. 10.

Strikingly, 50 % of all paths experience no stretch at all except for the slightly outlying AS 1239 topology. Larger stretches exceeding two are very rare—mainly in less than 10 %. These results are tightly connected

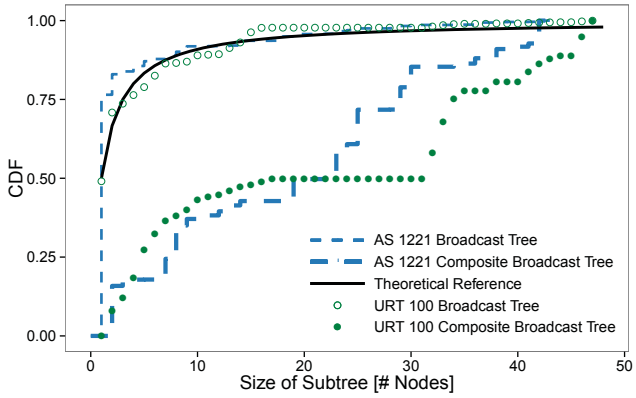


Figure 9. Distribution of subtree sizes for single and composite broadcast trees at FIB size 50—comparison of measurements from two topologies with theory

with the placement of NACs. Being at the center of the network, many shortest paths traverse through the NAC and experience no stretch. Caching at NAC will improve the results even further.

V. Conclusions and Outlook

Name based routing and forwarding in Information Centric Networking offer interesting potentials, but continue to raise significant challenges. In this work, we proposed a way to limit routing table sizes and to benefit from name aggregation within topological constraints. We introduced and thoroughly analysed PANINI, which may lead a new way to simplified content networking. Experimental as well as a theoretical evaluations revealed promising results in several dimensions.

In summary, we could show that PANINI routing is a self-optimizing hybrid approach that mitigates between FIB sizes and Interest flooding while locating content. Evidence was presented that rigorously small, incomplete routing tables can be compensated by a negligible quantity

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

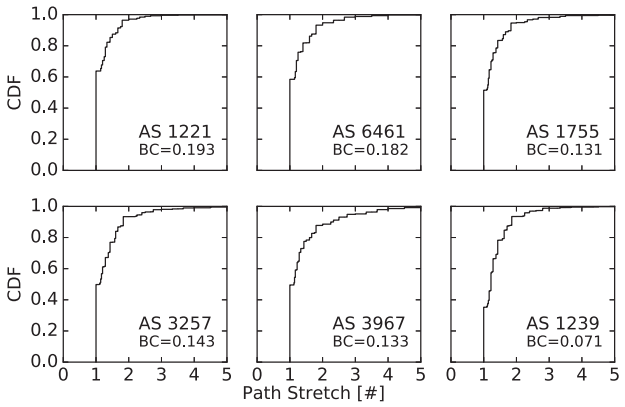


Figure 10. Distribution of path stretchs for PANINI Routing in different Rocketfuel topologies, BC denotes the Betweenness Centrality of the NAC

of broadcasts. Our future work will concentrate on to elaborate and quantitatively evaluate the aggregation potentials in distributed name-based routing. For the default free zone, this will raise the particular challenge of a scalable name synchronisation at interdomain peering. Corresponding routing strategies need to be found. It is our intend to show the feasibility of PANINI even for large-scale inter-provider settings.

References

[1] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "ICN Research Challenges," IETF, Internet-Draft – work in progress 06, March 2016.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, and M. F. Plass, "Networking Named Content," in *Proc. of the 5th Int. Conf. on emerging Networking EXperiments and Technologies (ACM CoNEXT'09)*. New York, NY, USA: ACM, Dec. 2009, pp. 1–12.

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, July 2012.

[4] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[5] T. C. Schmidt, S. Wölke, N. Berg, and M. Wählisch, "Partial Adaptive Name Information in ICN: PANINI Routing Limits FIB Table Sizes," in *2nd ACM Conference on Information-Centric Networking (ICN 2015), Poster Session*. New York: ACM, Oct. 2015, pp. 193–194.

[6] Y. Chung, "Distributed Denial of Service is a Scalability Problem," *SIGCOMM CCR.*, vol. 42, no. 1, pp. 69–71, Jan. 2012.

[7] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp, "Backscatter from the Data Plane – Threats to Stability and Security in Information-Centric Network Infrastructure," *Computer Networks*, vol. 57, no. 16, pp. 3192–3206, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2013.07.009

[8] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "MDHT: A Hierarchical Name Resolution Service for Information-centric Networks," in *Proc. of the ACM SIGCOMM WS on ICN*. New York, NY, USA: ACM, 2011, pp. 7–12.

[9] K. V. Katsaros, X. Vasilakos, T. Okwii, G. Xylomenos, G. Pavlou, and G. C. Polyzos, "On the Inter-domain Scalability of Route-by-Name Information-Centric Network Architectures," in *Proc. of IFIP Networking*, 2015.

[10] T. Song, H. Yuan, P. Crowley, and B. Zhang, "Scalable name-based packet forwarding: From millions to billions," in *Proc. of ACM ICN*. New York, NY, USA: ACM, 2015, pp. 19–28.

[11] J. J. Garcia-Luna-Aceves, "A More Scalable Approach to Content Centric Networking," in *Proc. of ICCCN*. Piscataway, NJ, USA: IEEE, 2015.

[12] A. Afanasyev, C. Yi, L. Wang, B. Zhang, and L. Zhang, "SNAMP: Secure Namespace Mapping to Scale NDN Forwarding," in *Proc. of IEEE Global Internet Symposium*. Piscataway, NJ, USA: IEEE, 2015, pp. 281–286.

[13] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "Pro-Diluvian: Understanding Scoped-Flooding for Content Discovery in Information-Centric Networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 9–18.

[14] R. Kleinberg, "Geographic Routing Using Hyperbolic Space," in *INFOCOM*. Piscataway, NJ, USA: IEEE Press, 2007, pp. 1902–1909.

[15] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá, "Hyperbolic Geometry of Complex Networks," *Physical Review E*, vol. 82, no. 036106, Oct 2010.

[16] D. Papadimitriou, D. Colle, P. Audenaert, and P. Demeester, "Geometric Information Routing," in *Proc. of IEEE ANTS*. Piscataway, NJ, USA: IEEE, 2013.

[17] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," *World Wide Web*, vol. 2, pp. 15–28, 1999, special Issue on Characterization and Performance Evaluation.

[18] M. Halvey, M. T. Keane, and B. Smyth, "Mobile Web Surfing is the Same as Web Surfing," *Commun. ACM*, vol. 49, no. 3, pp. 76–81, 2006.

[19] G. K. Zipf, "Relative Frequency as a Determinant of Phonetic Change," *Harvard Studies in Classical Philology*, vol. 40, pp. 1–95, 1929.

[20] P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad, "A Scaling Law for the Hopcount in Internet," Delft University of Technology, Tech. Rep., 2000.

[21] P. Van Mieghem, *Performance Analysis of Communications Networks and Systems*. Cambridge, New York: Cambridge University Press, 2006.

[22] M. Drmota, *Random Trees: An Interplay between Combinatorics and Probability*. Springer Science & Business Media, 2010.

[23] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF, RFC 6550, March 2012.

[24] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information Centric Networking in the IoT: Experiments with NDN in the Wild," in *Proc. of 1st ACM Conf. on Information-Centric Networking (ICN-2014)*. New York: ACM, September 2014, pp. 77–86. [Online]. Available: http://dx.doi.org/10.1145/2660129.2660144

[25] T. C. Schmidt, M. Wählisch, and S. Krishnan, "Base Deployment for Multicast Listener Support in Proxy Mobile IPv6 (PMIPv6) Domains," IETF, RFC 6224, April 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6224.txt

[26] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)," IETF, RFC 5015, October 2007.

[27] Q. Feng, C. Su, and Z. Hu, "Branching Structure of Uniform Recursive Trees," *Science in China Series A: Mathematics*, vol. 48, no. 6, pp. 769–784, 2005.

[28] N. Johnson and S. Kotz, "Urn Models and Their Application: An Approach to Modern Discrete Probability Theory," 1977.

[29] A. Panholzer and H. Prodinger, "Level of Nodes in Increasing Trees Revisited," *Random Structures & Algorithms*, vol. 31, no. 2, pp. 203–226, 2007.

[30] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible Network Experiments Using Container-based Emulation," in *Proc. of CoNEXT '12*. New York, NY, USA: ACM, Dec. 2012, pp. 253–264.

[31] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies With Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, 2004.

Scalable URL Matching with Small Memory Footprint

Anat Bremler-Barr[†], David Hay[*], Daniel Krauthgamer[†], and Shimrit Tzur-David[‡]
[†]Dept. of Computer Science, the Interdisciplinary Center Herzliya, Israel. {bremler,krauthgamer.daniel}@idc.ac.il
[*]School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel. dhay@cs.huji.ac.il
[‡]Dept. of Computer Science, Ben-Gurion University of the Negev, Israel. tzurdavi@cs.bgu.ac.il

***Abstract*— URL matching lies at the core of many networking applications and Information Centric Networking architectures. For example, URL matching is extensively used by Layer 7 switches, ICN/NDN routers, load balancers, and security devices. Modern URL matching is done by maintaining a rich database that consists of tens of millions of URL which are classified to dozens of categories (or egress ports). In real-time, any input URL has to be searched in this database to find the corresponding category.**

In this paper, we introduce a generic framework for accurate URL matching (namely, no false positives or miscategorization) that aims to reduce the overall *memory footprint*, while still having low matching latency. We introduce a dictionary-based compression method that compresses the database by 60%, while having only a slight overhead in time. Our framework is very flexible and it allows hot-updates, cloud-based deployments, and can deal with strings that are not URLs.

I. Introduction

As networks become more application- and service-oriented, *URL matching* is becoming an important component in many network devices and middleboxes.

In particular, URL matching is the basic building block of *layer 7 switches, routers, and load balancers* [18], [20], [25], [36], where routing decisions (e.g., which egress to forward a packet) are often determined by a URL (or a name) within a header of some application-layer protocol. Thus, URL (or, alternatively, service name or any other hierarchical human-readable names) matching is extensively used under Content-Centric Networking (CCN) approaches, such as Service-Centric Naming (e.g., Serval [20]) and Information Centric Networking [13] architectures like *Named Data Networking* (NDN) [2], [14], [38]. In particular, forwarding tables in high-speed NDN routers are expected to hold between $1-10$ millions URLs.

URL matching is also important in traditional networking, where it is primarily used to *enforce usage or security policy*. Modern security devices, especially in enterprises and workplace networks, are now filtering web content according to URLs (see Checkpoint [16], Palo Alto Networks [21], WebSense [33], Sourcefire [26], and others [28], [29]). In the past, such URL filtering consisted only on two categories: a blacklist of URLs that cannot be accessed and a white-list of legitimate URLs. However, contemporary URL filtering tools support tens of categories, allowing fine-grained policies which can be easily customized. Today, URL filtering tools have $1-100$ million URLs in these lists. URL matching is also used for other applications such as URL shortening services [27], [37], and search engines [15].

ISBN 978-3-901882-83-8 © 2016 IFIP

As the average length of a URL is 22.6 bytes, the memory footprint of the URL database often becomes humongous. On the other hand, URL matching is often performed as a *bump on the wire*, implying the URL database must be stored in an expensive fast memory to support line-rate queries. Thus, compressing the database while obtaining fast queries is essential, either to make URL-matching–based solutions feasible, or to significantly reduce their costs.

This paper tackles exactly this problem and presents a generic framework to efficiently store URLs in a *database*, along with their *categories* (in the context of Layer 7 routing or NDN/ICN, the database corresponds to a forwarding table/FIB, and a category corresponds to an egress port or ports). When the database is queried, it either returns the category attached to the queried URL or $\perp$ in case the URL is not in the database. Importantly, *we do not allow inaccurate results*—the query must always return the correct answer.

Our approach is generic in the sense that it does not have any assumption on the data structure used to perform the URL matching itself (This data structure is referred to as *database* in Fig. 1). Our framework compresses only the *information* stored in that data structure, and therefore, can leverage from any fast URL matching technique (called *database query* in Fig. 1). Specifically, our compression can be performed on the entire URL at once (applicable mainly for hash-based solution) or in a component-by-component manner (applicable, for example, to Trie-based solutions).

This paper focuses on *reducing the memory footprint of the database, while still enabling fast queries.* Since URLs share many common substrings, a naïve approach would

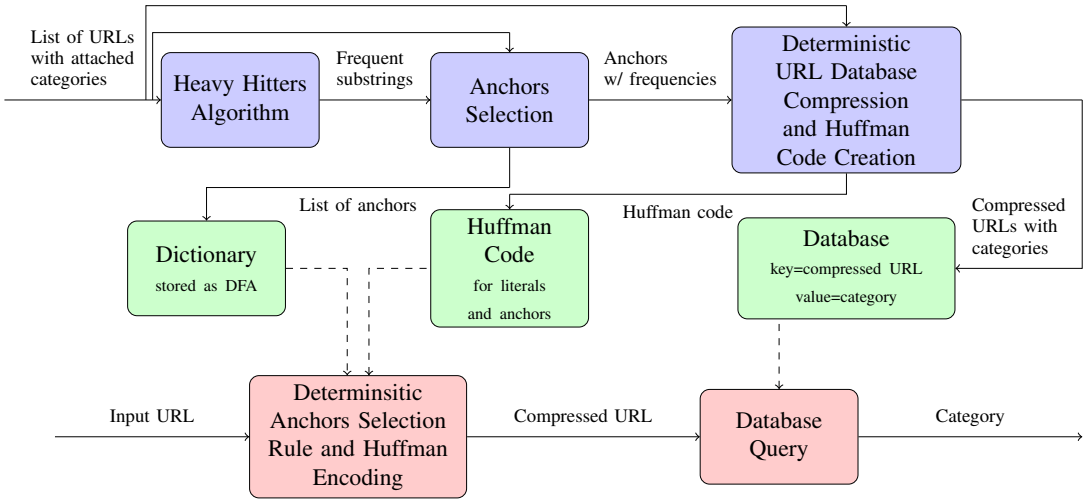


Fig. 1. A block diagram of our framework. The blue blocks describe the offline phase, where the database and auxilary data structures are built. The red blocks are the datapath of our framework. Green blocks represent the data structures, which constitute the memory footprint we try to minimize.

have been to compress the database using an off-the-shelf compression algorithm (e.g., DEFLATE used by gzip [9]). Indeed, such compression reduces the memory footprint by about 70%, however it does not allow fast queries of the compressed database. This essentially stems from the fact that the compressed form of each individual URL depends on previous URLs in the database.

Thus, we take a different approach and use a *dictionary-based compression*, which is illustrated in Figure 1. Our framework is divided to two phases: an offline phase, in which the database is built, and a datapath, in which input URLs are queried against the database.

In the offline phase, we start by extracting the frequent substrings[1] that appears in the URLs; (for example, the three most such substrings in our data were "`.com`", "`s.com`", and "`e.com`"); this is done by off-the-shelf heavy hitters algorithms, such as [1], [10]. Notice that frequent substrings intersect each others, and therefore, sometimes a substring may not be useful for compression, even though it appears many times. Thus, for each frequent substring, we first estimate its real frequency in the compressed database, and then try to determine whether it is indeed beneficial to use it (taking into account, for example, also its length and the overhead in storing it in the dictionary). Using these estimates, we select a subset of the frequent substrings as *anchors*, which will be stored in the dictionary. We also use the estimated frequency of the anchors to create a Huffman code for them (which will further reduce the memory footprint). We note that given the Huffman code and the dictionary, we can compress each URL separately, by replacing each anchor (and each *literal*) by the corresponding Huffman code. All compressed URLs are stored along with their categories in an off-the-shelf database (e.g., based on a hash-table [35], [37] or a Trie). We note that sometimes more than one anchor replacement option is available (e.g., suppose our dictionary is §1="`goo`", §2="`.com`", and §3="`ogle`"; the URL `google.com` can be compressed to either §1`gle`§2 or `go`§3§2). Hence, our compression algorithm uses a *deterministic rule* to choose the proper replacement, implying that the compressed form of each URL depends only on the dictionary used.

The datapath works in a similar way: We first identify which anchors are in the input URLs, then we use the same deterministic rule to choose which anchors to use, and finally we use the same Huffman code to encode the selected anchors and literals. Thus, when querying the database, it is sufficient to use the compressed-form of the queried URL as a key to the database.

Overall our framework yields a reduction of up to 60% in memory. Naturally, using compression trades memory space with the overhead of processing (compressing) the data. However, our experiments shows that the processing overhead (namely, the URL compression) works at rate of more than 200 Mb/s on one core, which is acceptable in these settings, as these operations can be highly paralleled (either by compressing many URLs on simultaneously on multiple cores, or by having the URL compression and database query as successive stages in a pipeline). In addition, we note that our framework allows *hot-updates support*, where URLs can be easily inserted, deleted, or modified in the database. In addition, we are able to deal with both *exact matching* and *longest prefix matching (LPM)* of URLs, where the only difference is that under LPM, we must first break the URL to its components (separated by dots or slashes) and then compress each

[1]We will use the terms *string* and *substring* interchangeably when it is clear from the context.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 468

such component separately.

We provide a full implementation of our framework in [6].

The rest of this paper is organized as follows: Section II discusses previous works on URL matching. In Section III, we describe the datapath of our framework, in which we resolve online the category of input URLs. Section IV describes the offline phase of our framework, in which the database and auxiliary data structures are constructed. In Section V, we present experimental results on real-life URL data sets. Finally, we conclude in Section VI.

II. Related Work

Performing fast URL matching has recently received extensive attention [2], [8], [30]–[32], mostly focusing on the URL matching *time* This paper focuses on reducing the memory footprint of the URL database by designing a tailored-made compression method. Once the database is compressed, it can be used in conjunction with any of the previously suggested URL matching techniques.

As opposed to our framework, which employs *lossless compression*, most of the previous works on URL compression have used lossy compression, such as a hash-chain compression [18], CRC compression [39], or hierarchical Bloom filters [11]. Clearly, lossy compression techniques have two clear drawbacks. First and foremost, they might lead to a false categorization of URLs, as several URLs (each with different category) might be compressed to the same form. In addition, these compressions are not reversible, implying updates might become more difficult.

The works in [11], [35], [37], [39] dealt only with two categories of URLs (blacklist and white-list), and cannot be easily extended to multiple categories (e.g., by assuming that each URL that is not in one category, is implicitly in the other). Hence, these works are not applicable to modern devices that have many categories and have to store significantly more URLs.

URL lossless compression using AVL tree was considered in [15]. This compression achieves 50% compression ratio, albeit with high query and insertion time (namely, tens of microseconds). We note that this solution aims to compress crawling information of web spiders, which is less time-sensitive. However, in our case, query time is important, as most URL matching lie in the critical path of the traffic. In other words, the required solution has to represent the data using as little space as possible, yet efficiently answering queries on the represented data.

Moreover, nowadays there are many URL shortening services such as [4], [27]. These services store a database that translates the shortened version of the URL back to its original form, and then redirects the user requests to the original address. Thus, this shortening is basically a *renaming* of the URLs rather than compression; the database holds the URLs in an uncompressed form and its total space is not reduced, and therefore, it is not applicable in our setting.

Another common strategy to deal with the string retrieval problem is using tries. However, tries-based solutions do not supply efficient solutions to a non-prefix query, unless a very large amount of memory is used. Note that providing a solution to the string retrieval problem usually requires to create an index to the data and the footprint that is required to represent this index is sometimes larger than the footprint of the original data. Compressing trie-based data structure for URL matching was considered in [8], [30]. In these works, the edges of the trie correspond to the components of the URLs and the basic idea is to provide each such component with a unique code; moreover, the same code can be used for different parts of the trie when its decoding is not ambiguous. It is important to notice that the technique compresses the trie-based data-structure, but its memory footprint is larger than the size of a bare list of all URLs, as almost the entire URL information is stored explicitly in the lookup table for finding the code of each component. Our framework, on the other hand, reduces the memory footprint below the size of that list, and can be used in conjunction with the techniques of [8], [30] to further reduce the memory footprint of the trie. Moreover, while the compression techniques of [8], [30] is specific to their trie-based data structure, our framework is generic and can be applied with any matching data-structure.

Usually, solutions that create an index of the data are heavy in space (e.g. [17]). Dealing with this inefficient footprint was a subject recent research (e.g., [12]). Our solution obtains better results, in compression ratio as well as lookup performance. Actually, the lookup performance of this work is two orders of magnitude slower than our solution, as presented at Section V. We also note that compressing string dictionaries was a subject for extensive research (c.f. [7], [19] and references therein); however, these solutions were not geared specifically to URL databases.

Retravi et al. have studied how to compress IP forwarding table [23]. However their solution is not applicable to URL matching, since it assumes either a limited number of categories (corresponding to the possible next hops) or very rare updates.

Finally, femtoZip [10] is a compression library that aims in compressing short documents. As our framework, it constructs a shared dictionary which are used by all documents; unlike our scheme, this shared dictionary is a *concatenation* of all anchors, and therefore, references to this dictionary is done by referring to the offset of the anchor from the beginning of the dictionary and the length of the anchor. Concatenating anchors yields a more compact dictionary (e.g., one can refer to all substrings of

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 469

each anchor, by the appropriate offset and length), yet references the dictionary from the compressed document require three bytes for 64 KB dictionary with anchors of up to 256 bytes. femtoZip is not applicable to our setting as, on one hand, it does not provide fast compression after the dictionary is built (needed for our datapath), and, even more importantly, holding the dictionary as a concatenation of anchors rules out efficient encoding of the anchors. As we will show later, Huffman encoding of the anchors achieves significant improvement in the compression ratio, as, on average, each anchor requires less than three bytes (this is due to the fact that there are many frequent anchors whose short encoding reduce the overall size significantly). Nevertheless, we use components of femtoZip as one of the alternatives for a heavy hitters algorithm (to extract frequent substrings), as described in Section IV-A.

III. The Datapath and Data Structures

In this section, we explain the different steps taken to resolve the category of an input URL. Essentially, it include a *compression stage*, and then a *database query* with the URL in a compressed-form as a key. Notice that this process is done online, and it assumes that the database and auxiliary data structures are given (see Figure 1). We will explain how to obtain the database and construct the data structures in Section IV. Moreover, in case that either a trie-based database is used or a longest-prefix matching is required, we first break the URL into components (by "`.`" and "`/`" delimiters) and then compress each component separately. First, given the input URL, we extract the anchors which are contained in that URL. This is done by applying a *pattern matching algorithm* on the URL, where the set of patterns is the set of anchors. In this work, we chose to use the Aho-Corasick algorithm [3], which is based on traversing a Deterministic Finite Automaton (DFA) representing the set of anchors. Thus, the dictionary is stored as a DFA, whose accepting states represent anchors. We note that it is useful to store additional information about each anchor in the DFA, namely the length of the corresponding Huffman code and a pointer to the Huffman code itself. Compressing the size of the DFA for pattern matching algorithm was the subject of an extensive research recently; in this work, we use the most compressed form as presented in [5].

We notice that at each byte traversal of the DFA, the *DFA state* represents the set of anchors which are suffixes of the URL up until the scanned byte. We will need to decide *deterministically* which of the anchors in this set should be used indeed for compression (recall the example in Section I where the URL `google.com` can be compressed into two different forms by different choice of anchors). As we aim to minimize the total length of the compressed URL, we are using the following greedy approach, which traverses the DFA one byte at a time, and (conditionally) pick the anchor that minimize the length of the scanned prefix.

Specifically, for each anchor or literal a, let $\ell(a)$ be its length in bytes and $p(a)$ be its Huffman code length. Let u_i be the i-th byte of the URL, and let S_i be the set of anchors which are suffixes of the prefix of the URL up until u_i (as represented by the DFA state after scanning u_i).

The *deterministic selection rule* works iteratively by maintaining two vectors P and V, such that $P[i]$ is the minimum length for encoding the first i bytes of the URL, and $V[i]$ is the last anchor or literal that achieves encoding length $P[i]$. Hence, for each byte i, we first calculate

$$V[i] = \underset{a \in S_i \cup \{u_i\}}{\arg\min} \left(P[i - \ell(a)] + p(a) \right),$$

and $P[i] = P[i - \ell(a_i)] + p(V[i])$ ($P[0]$ is always 0).

When completing the traversal of the entire URL, we go backwards on the elements of $V[i]$ and concatenate them, skipping non-useful elements (namely, after adding $V[i]$, we add element $V[i - \ell(V[i])]$, skipping all elements in between). It is easy to verify by induction that this selection results in an optimal-length encoding (given the set of anchors and Huffman code), thus achieving the best compression ratio.

Fig. 2 depicts a step-by-step example of compressing the URL `comrgnetwork.com` in a component-by-component manner. Note that, for example, in the 5th step, the DFA finds a matching with anchor `mrg`; however the value of $P[5 - \ell(\texttt{mrg})] + p(\texttt{mrg}) = P[2] + 3 = 14$ is larger than choosing the last literal `g`. When scanning the arrays backwards, the anchors and the literals `network`, `g`, `r`, and `com` are selected. The second component `com` is compressed by running the first three steps of the above-mentioned execution.

The final step is to use the compressed-form to query the database. Since we require accurate results (namely, no false positive and miss categorizations), the database maintains also the compressed-form of the URL and not only its category. Most current implementation uses a hash-table to maintain the database [37]. Thus, by comparing the lookup key with the stored key, one avoids miss-categorization due to hash collisions. Trie-based solutions and longest-prefix matching usually require a component-by-component compression and lookup. Clearly, our framework readily supports such data-structures and matching, albeit with smaller compression ratio as some compression opportunities (e.g., anchors that span more than one component) may be missed.

IV. The Offline phase: Building the Database

As illustrated in Figure 1, the offline phase of our framework, consists of three steps:

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 470

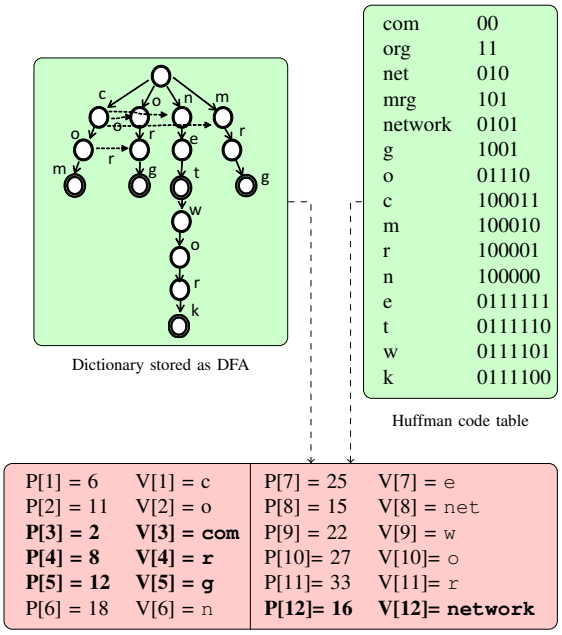


Fig. 2. Compressing the URL `comrgnetwork.com` in a component-by-component manner. As shown in Fig. 1, the compression uses a dictionary stored as a DFA and a corresponding Huffman code table. The resulting compressed URL comprises of the anchors `com,r,g,network,com` and is of length 18 bits (16 bits for the first component whose code is 0010000110010101 and 2 bits for the second component whose code is 00).

Step 1: **Heavy hitters algorithm**, in which we find a set of k frequent substrings in the set of URLs database.

Step 2: **Anchors selection**, in which we pick, from the frequent substrings, a final set of anchors. For each anchor and literal, we also calculate the estimated number of occurrences in the compressed URL database

Step 3: **Deterministic URL database compression and Huffman code creation**, in which we use the selected anchors to replace substrings in each URL separately. We also create an Huffman code using the given frequencies of literals and anchors, which we then use to encode each URL.

Next we elaborate on each step.

A. Heavy Hitters Algorithm

We compare between two off-the-shelf alternatives to extract the most frequent substrings in the list of URLs.

The algorithm described in [1] is geared to find frequent substrings of variable length. Specifically, it returns all the substrings which have unique frequency larger than n/k, where n is the number of URLs and k is a parameter that aims to calibrate the number of frequent substrings we need to find. The algorithm is approximated and the frequency of each substring is only estimated (with an error bound of $3n/k$). Note that, in any case, this frequency is not used later by our algorithm, as subsequent steps will estimate the actual number of times each substring is used for compression.

This algorithm works in time complexity of $O(n \cdot L)$, where L is the average URL length, in $O(k \cdot \ell)$ space complexity. The algorithm requires only one pass on the URL database and its space requirement is proportional to the number of heavy hitters. Yet, the results are only approximated with small error in the estimated frequency of the substrings and thus the algorithm might not find the real k frequent substrings.

Moreover, we note that this algorithm avoids space pollution and if a substring s is selected as a heavy hitter then the algorithm would not count appearances of a substring s', such that $s' \subseteq s$, when s' is within s. Nevertheless, appearances of s' not within s are counted, and if s' appears frequently alone, it might be selected as a heavy hitter together with s. See, for example, Fig. 3 that presents component-by-component compression of URLs. The heavy-hitters algorithm with $k = 5$ picks the substrings "`network`" and "`net`" as anchors, but not the substring "`netwo`" that never appears by itself. Naturally, when processing the entire URLs at once (see Fig. 4) the heavy-hitters algorithm finds longer anchors such as "`network.com`".

The second alternative is to use the heavy-hitters algorithm of femtoZip library. Unlike [1], this heavy-hitters algorithm is accurate, and it works by essentially enumerating all the possible substrings. Thus, it is significantly more time and space intensive, with time complexity of $O(n \cdot L \cdot \log(n \cdot L))$ and space complexity of $O(n \cdot L)$.

While we are less concern with the performance of the components in the offline phase, it is still desirable to reduce them as much as possible. In the experimental section, we show that, in practice, the compression ratio stays almost the same, whether we use the accurate or the approximate algorithm.

B. Anchors Selection

As explained before, the fact that frequent substrings intersect implies that a substring might not be used to compress sufficiently many URLs, even though it is frequent. Yet, these substrings increase the size of the dictionary, and therefore, should be eliminated.

Thus, in this step, we pick *anchors* out of the frequent substrings. Specifically, we first estimate, for each frequent substring, the *database compression frequency*—the number of times it will be used in the database compression (which is smaller than the frequency attached to it by the heavy hitter algorithm). Then, based on this estimated frequency, we will approximate both the gain in selecting the substring and the loss in terms of dictionary size, so that each substring whose gain is larger than its loss will be selected as an anchor. Finally, given the definitive

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 471

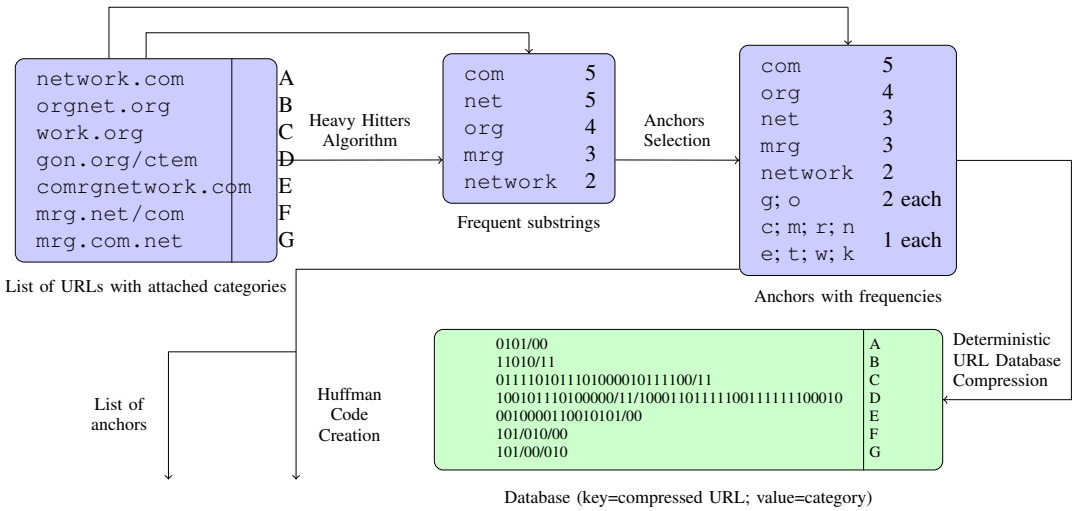


Fig. 3. An example of the offline phase of the algorithm, where compression is done in a component-by-component manner. In the compressed database, we use "/" as a delimiter between components; this delimiter is either used within a trie-based data structure or should be encoded separately. The corresponding auxiliary data structures used by the datapath (namely, the dictionary as a DFA and the Huffman code table) appear in Fig. 2.

selection of anchors, we adjust the frequency of both anchors and literals.

1) Estimating the database compression frequency of a substring: In order to calculate the estimated compression frequency, we try to estimate the compression process, as explained in Section III. Since, at this point, we cannot know what will be the Huffman code of each anchor or literal, we assume in this phase that the length of encoding each literal and each anchor is the same and, without loss of generality, is set to 1. This implies that the length of a compressed URL is estimated by the sum of the number of anchors in the compressed URL and the number of the remaining literals (e.g., in the example given in Section I, the estimated length of §1gle§2 is 5 and the estimated length of go§3§2 is 4).

As explained in I, a single URL compression involves a *deterministic selection rule* of specific anchors out of a larger set of anchors. In this step, we apply the same rule to select anchors out of the set of frequent strings, which implies we build a *temporary* DFA for all frequent substrings, set $p(a) = 1$ for each literal and anchor a, and run the greedy algorithm of Section III, one URL at the time, for all URLs in the list. Each time a frequent substring is selected as an anchor when compressing a single URL, we increase the substring's database compression frequency by 1. In the end of the process, we will have an estimation of the database compression frequencies of each substring. Notice that this is just an estimation, since not all substrings (with a frequency of at least 1) will be selected as anchors, implying the deterministic selection rule in Step 3 might yield different results. In addition, another difference in selection might be as a results of variable length encoding (with Huffman code) in Step 3; see Fig. 3 that also illustrates the new calculated frequency, where the frequency of the substring "net" is reduced as in two of its appearances the substring "network" was selected. A sketch of the DFA representing the DFA appears in Fig. 2, where some of the edges are omitted for clarity; accepting states are marked with double circles.

2) Selecting anchors out of frequent substrings: We note that while replacing a parts of a URL by anchors reduces the URL size, it comes with a price: each anchor increases the size of the dictionary's DFA and, in addition, the anchor's encoding needed to be tracked, implying even further memory footprint. Therefore, we need to avoid picking up substrings that are not used sufficiently many times.

Let A be the set of all frequent substrings, let Σ be the set of all literals, and let $f(a)$ be the number of times substring $a \in A \cup \Sigma$ was used in the previous step. If an anchor $a \in A$ is selected, for each of these $f(a)$ times, we save $\ell(a) - \text{huffman}(a)$ bytes, where $\ell(a)$ is the length of a in bytes and $\text{huffman}(a)$ is the length of the Huffman code of a. Since we cannot calculate the Huffman code of a yet, we estimate it by anchor a's *information content*:

$$h(a) = \frac{1}{8} \log_2 \frac{\sum_{a \in A \cup \Sigma} f(a)}{f(a)}.$$

Note that Huffman code strives to encode each anchor a with $h(a)$ bits. Thus, the total *gain* of selecting a is $f(a)\,(\ell(a) - h(a))$.

On the other hand, inserting a to the data structures, requires adding states to the DFA and storing its Huffman code. As explained before, we estimate the Huffman code cost by $h(a)$. As described in [5], the footprint of the DFA in its compressed form, is approximately $C_{state} = 4$ bytes

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 472

per state. Notice that two anchors that share a common prefix, share also common states in the DFA. Hence, we use the following procedure to decide whether a substring is selected as an anchor. We first have an empty DFA, and sort the substrings in descending order of their *gain*. For each frequent substring $a \in A$ in turn, we calculate the number of states $states(a)$ it requires (on top of the existing DFA) and $h(a)$. If $f(a)\,(\ell(a) - h(a)) \geq C_{state} \cdot states(a) + h(a)$, we select a as an anchor, update the DFA, and continue to the next substring. Otherwise, we leave the DFA unchanged and skip substring a. In the end of the process, we will have the set of all anchors and the corresponding DFA, that is used in the datapath.

3) Re-estimating the frequency of anchors and literals: Since only a subset of the frequent strings was selected as anchors, the frequency of anchors and literals can be changed significantly. Thus, we ran the greedy algorithm of Section III, using the DFA that was created in Section IV-B2 and with $p(a) = h(a)$ for each literal and each anchor a. This will result in an updated frequency estimation of each anchor and literal.

C. Deterministic URL database compression and Huffman code creation

Now that we have the anchors and their estimated frequency, as well as the estimated frequency of all literals, we construct the Huffman codes in a standard way, treating all anchors and literals as symbols (and, thus, ignoring their original size). The result is stored in the Huffman code data structure (namely, a table with entry for each literal and anchor, where the entries of anchors are pointed out by the corresponding DFA state). We then run once again the algorithm of Section III with the correct $p(a)$ value for each anchor and literal a. This will results in compressing each URL separately. Each compressed URL will be then inserted into the database along with its category; see Fig. 3.

D. Hot-updates Support

In order to insert a new URL, we first obtain its compressed form by going over all the steps of the datapath. Then, instead of querying the database, we perform an `insert` operation with the compressed-form URL as a key and the category as a value. Similar operation should be done in order to update a category of a URL.

Periodically, the algorithm can rebuild the database from scratch, as the frequency of substrings might change over time, resulting in suboptimal encoding.

V. Experimental Results

We have used an open-source database of URLs available in `URLBlackList.com` [28]. This daily-generated list consists of $2,200,000$ unique domain names and 95 different categories. We note that than $800,000$ URLs have also paths and not just domain names.

Compression Method	*Compression Ratio*
femtoZip	0.57
Huffman encoding only	0.59
Our Framework (accurate heavy hitter)	0.43
Our Framework (approximate heavy hitter)	0.44

TABLE I
COMPARISON BETWEEN THE COMPRESSION RATIOS OF DIFFERENT METHODS FOR MODERATE SIZE DATABASE OF 128,000 DOMAIN NAMES AND 128,000 URL COMPONENTS. THE SIZE OF THE DATABASE IN AN UNCOMPRESSED FORM IS 57.2 MB, WHILE OUR FRAMEWORK COMPRESSES THE DATABASE UP TO 24.3 MB.

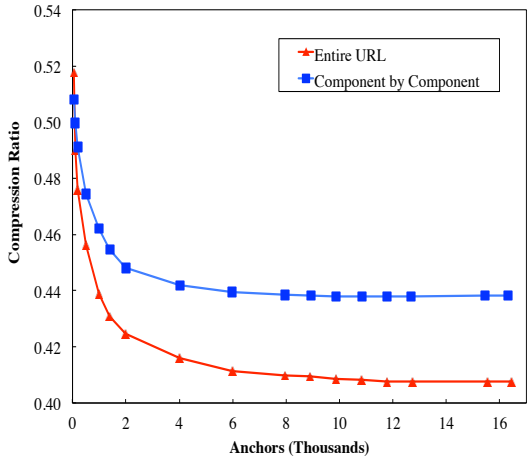


Fig. 5. The effect of the number of anchors used on the compression ratio. 50 anchors achieve 52% compression ratio for entire URLs, and 49% compression ratio where the URL is compressed component-by-component. With 16K anchors, the compression ratio improves to 40.77% and 43.8%, respectively.

The focus of this paper is the memory footprint of our framework (namely, the size of the database in its compressed-form and the size of the auxiliary data structures). This is captured by the *compression ratio*, which is the ratio between its memory footprint and the size of the database with uncompressed URLs. Notice that smaller compression ratio is better. More specifically, we have calculated our memory footprint by summing up the size of the dictionary (represented as DFA), the size of the Huffman code table, and the length of each URL in the database in its compressed form. We compare this footprint with the total length of all URLs in their uncompressed form.

We note that the memory footprint of the auxiliary data structures is only $0.1\% - 3.3\%$ of the overall memory footprint (the exact percentage depends on the number of anchors used). In practice, padding and fragmentation issues may increase this memory footprint significantly. Therefore, we have designed and implemented tailored-made memory allocator that reduces the overhead to 30%, implying that, in practice, at most 4.2% of the memory footprint is used for the auxiliary data structures.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 473

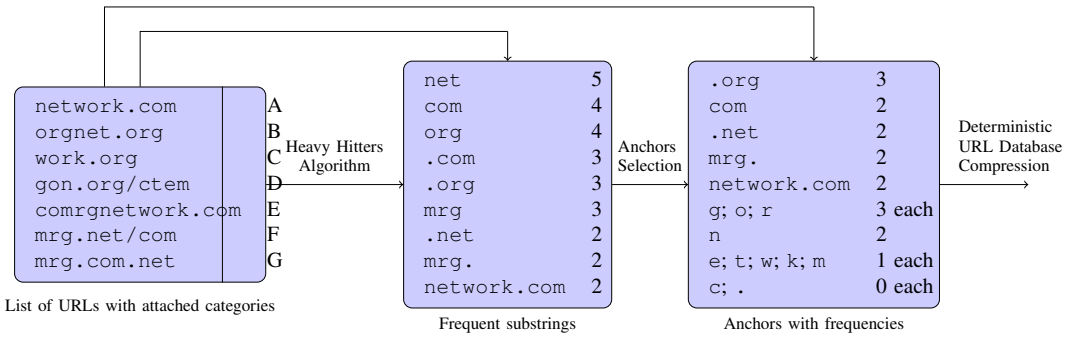


Fig. 4. An example of the offline phase of the algorithm, where compression is done in on the entire URL at once. In this example, the heavy hitter algorithm is configured with $k = 9$ and we consider only the domain part of the URLs. We note that anchors that are left with one appearance are not selected, and some literals always appear as part of one of the anchors.

A. Comparison with other lossless compression methods

Table I shows the compression ratio for several compression methods. We note that as we need to maintain low matching latency, we must compress each URL separately. Thus, methods that use backward-references (namely, applying either zip [22], bzip2 [24], or lzw [34], [40], on each URL separately) are not useful in this case (and in fact, even increase the size of it due to overhead of information they store for compression). As expected, femtoZip [10], which aims in compressing short strings, achieves reasonable compression ratio when the number of URLs is sufficiently large. Yet, it lacks an efficient datapath (namely, after the database compression, only decompression of URL is easy, while compression of new URLs using the same dictionary is difficult). Encoding the literals of the URLs with Huffman codes also reduces the memory footprint by approximately the same factor. Nevertheless, our framework outperforms all other methods. There are negligible difference between the performance of our framework with accurate or approximate heavy hitter algorithm.

B. The throughput of the datapath compression

We have implemented our datapath in C and used Intel Xeon E5-2690V3 CPU, whose processor speed is 2.6 GHz with 16 GB memory (per core), L2 cache of 256 KB (per core), and L3 cache of 30 MB. The system runs Ubuntu 14.04.2 LTS.

Each performance number was measured by applying the datapath compression 20 times on 10 randomly-selected sets of 10,000 URLs (namely, 200 runs per performance number, each representing compression of 10,000 URLs in a batch). All our experiments ran only on a *single* core. We did not measure the data-base lookup as this is orthogonal to our framework and can be implemented as a successive pipeline stage.

Fig. 6 shows the throughput (per core) of the datapath as a function of the number of anchors. The performance

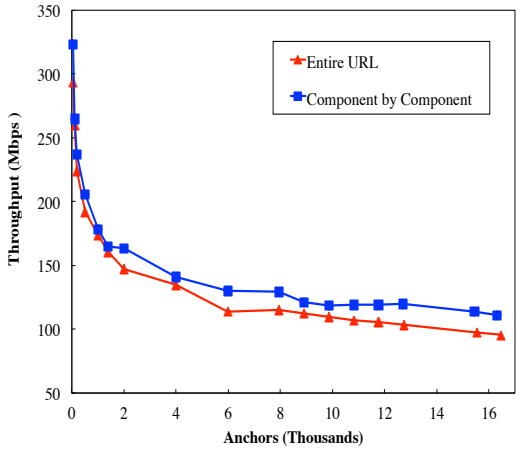


Fig. 6. The throughput of the URL compression stage in the datapath.

of the datapath depends both on the size of the auxiliary data structure and on the number of anchors (per URL) used for compression. As the number of anchors increases, longer anchors are added to the dictionary, implying the average number of anchors per URL decreases. However, larger number of anchors implies larger data structures that might not fit entirely in cache, thus causing performance degradation. In any case, the throughput is between 320 Mb/s to 100 Mb/s.

VI. Conclusions

This paper introduces a framework to significantly reduce the memory footprint of URL-based databases and forwarding tables, while maintaining the accuracy of the lookup processing (namely, no false positives or miscategorizations) and incurring only a small overhead in time. The framework also allows hot updates of the database and a longest prefix matchings.

We note that a common deployment of URL-matching–enabled devices is to store the rich database in a cloud

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 474

and store locally (in a cache) recently-matched URLs. Upon a cache-miss, the networking device will query the database in the cloud for the correct category of a URL. Our framework can be deployed in both locations. In this setting, the databases will be stored in a compressed form in the cloud, and upon a cache-miss, the networking device will compress the input URL and use the compressed form to query the database. This implies that all traffic between the networking device, its cache (if applicable), the cloud, and intra-cloud communication is done with the compressed URL, whose size is only approximately 40% of the uncompressed one. Naturally, the latency of the URL matching processes is dominated by the latency between the security tool and the cloud (the processing overhead is negligible). This implies that it may be beneficial to store in the cache also compressed URLs, increasing their number by a factor of 2.5.

Acknowledgments: This research was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement n^{o} 259085, and also by the Neptune Consortium, administered by the Office of the Chief Scientist of the Israeli ministry of Industry and Trade and Labor.

REFERENCES

[1] Y. Afek, A. Bremler-Barr, and S. Landau Feibish. Automated signature extraction for high volume attacks. In *ACM/IEEE ANCS*, 2013.

[2] R. Ahmed, F. Bari, S.R. Chowdhury, G. Rabbani, R. Boutaba, and B. Mathieu. αroute: A name based routing scheme for information centric networks. In *IEEE INFOCOM*, pages 90–94, 2013.

[3] AV. Aho and MJ. Corasick. Efficient string matching: an aid to bibliographic search. *Commun. of the ACM*, pages 333–340, 1975.

[4] bitly, Inc., 2013. `https://bitly.com`.

[5] A. Bremler-Barr, Y. Harchol, and D. Hay. Space-time tradeoffs in software-based deep packet inspection. In *IEEE HPSR*, 2011.

[6] Anat Bremler-Barr, David Hay, Daniel Krauthgamer, and Shimrit Tzur-David, 2015. `https://github.com/DeepnessLab/urlmatching`.

[7] N.R. Brisaboa, R. Cánovas, F. Claude, M. Martínez-Prieto A, and G. Navarro. Compressed string dictionaries. In *Experimental Algorithms*, pages 136–147. 2011.

[8] H. Dai, B. Liu, Y. Chen, and Yi Y. Wang. On pending interest table in named data networking. In *ACM/IEEE ANCS*, pages 211–222, 2012.

[9] P. Deutsch. GZIP file format specification version 4.3. IETF RFC 1952, 1996.

[10] femtoZip. Shared dictionary compression library, 2013. `https://github.com/gtoubassi/femtozip/wiki`.

[11] Y.-H. Feng, N.-F. Huang, and C.-H. Chen. An Efficient Caching Mechanism for Network-Based URL Filtering by Multi-Level Counting Bloom Filters. In *IEEE ICC*, 2011.

[12] P. Ferragina and R. Venturini. The compressed permuterm index. *ACM Trans. Algorithms*, 7(1):10:1–10:21, December 2010.

[13] IRTF. Information-centric networking research group (icnrg). https://irtf.org/icnrg.

[14] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *ACM CoNEXT*, pages 1–12, 2009.

[15] K. Koht-Arsa and S. Sanguanpong. In-memory URL compression. In *National Computer Science and Engineering Conference*, 2001.

[16] Check Point Software Technologies LTD. Check point URL filtering software blade, 2013. `http://www.checkpoint.com/products/url-filtering--software-blade/`.

[17] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[18] B. Scott Michel, Konstantinos Nikoloudakis, Peter Reiher, and Lixia Zhang. URL Forwarding and Compression in Adaptive Web Caching. In *IEEE INFOCOMM*, pages 670–678, 2000.

[19] G. Navarro and V. Mäkinen. Compressed full-text indexes. *ACM Computing Surveys*, 39(1):2, 2007.

[20] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Ko, J. Rexford, and M. J. Freedman. Serval: an end-host stack for service-centric networking. In *USENIX NSDI*, 2012.

[21] Palo Alto Networks. Next-generation firewall features, 2013. `http://www.paloaltonetworks.com/products/features/-url-filtering.html`.

[22] PKWARE, Inc. zip, an archive file format, 1989. `http://www.pkware.com/support/zip-app-note/`.

[23] G. Retvari, J. Tapolcai, A. Korosi, A. Majdan, and Z. Heszberger. Compressing ip forwarding tables: Towards entropy bounds and beyond. *IEEE/ACM Transactions on Networking*, 2014.

[24] J. Seward. bzip2 and libbzip2, a program and library for data compression, 2007. `www.bzip.org`.

[25] W. So, A. Narayanan, D. Oran, and Y. Wang. Toward fast NDN software forwarding lookup engine based on hash tables. In *ACM/IEEE ANCS*, pages 85–86, 2012.

[26] Sourcefire, Inc. Intelligent cybersecurity solutions, 2013. `http://www.sourcefire.com/security-technologies/-network-security/next-generation-firewall`.

[27] TinyURL!^{TM}. Making over a billion long urls usable! serving billions of redirects per month., 2013. `http://tinyurl.com`.

[28] URL.Blacklist.com, 2013. `http://urlblacklist.com/`.

[29] URLfilterDB. URL filter for the Squid web proxy, 2013. `http://www.urlfilterdb.com/`.

[30] Y. Wang, K. He, H. Dai, W. Meng, J. Jiang, B. Liu, and Y. Chen. Scalable name lookup in ndn using effective name component encoding. In *IEEE ICDCS*, pages 688–697, 2012.

[31] Y. Wang, T. Pan, Z. Mi, H. Dai, X. Guo, T. Zhang, B. Liu, and Q. Dong. Namefilter: Achieving fast name lookup with low memory cost via applying two-stage bloom filters. In *IEEE INFOCOM*, pages 95–99, 2013.

[32] Y. Wang, Y. Zu, T. Zhang, K. Peng, Q. Dong, B. Liu, W. Meng, H. Dai, X. Tian, and Z. Xu. Wire speed name lookup: A gpu-based approach. In *USENIX NSDI*, pages 199–212, 2013.

[33] Websense, Inc. The Websense Master Database. `http://www.websense.com/content/urlcategories.aspx`.

[34] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, June 1984.

[35] T. Yamauchi, H. Yuan, and P. Crowley. Implementing URL-based forwarding on a network processor-based router platform. In *ACM/IEEE ANCS*, pages 171–172, 2009.

[36] H. Yuan, T. Song, and P. Crowley. Scalable NDN Forwarding: Concepts, Issues and Principles. In *IEEE ICCCN*, 2012.

[37] H. Yuan, B. Wun, and P. Crowley. Software-based implementations of updateable data structures for high-speed URL matching. In *ACM/IEEE ANCS*, page 15, 2010.

[38] Lixia Zhang et al. Named Data Networking Project (NDN), 2010.

[39] Z. Zhou, T. Song, and Y. Jia. A high-performance url lookup engine for url filtering systems. In *IEEE ICC*, pages 1–5, 2010.

[40] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory*, 24(5):530–536, 1978.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP 475

Network Quality Differentiation: Regional Effects, Market Entrance, and Empirical Testability

Toni Mäki
VTT Technical Research Centre of Finland
Oulu, Finland

Patrick Zwickl
University of Vienna
Faculty of Computer Science
Vienna, Austria

Martín Varela
VTT Technical Research Centre of Finland
Oulu, Finland

***Abstract*—While for offline business models it does not seem necessary to reiterate the close relationship between quality and price, for Internet services the quality-based, i.e., Quality of Experience (QoE), and customer-centric pricing is non-trivial. As insufficient data exists today to successfully commercialise QoE, this paper collects the integral empirical Willingness-To-Pay (WTP) data for the case of online video services. This work reproduces and extends a previous study in two dedicated campaigns in Austria and Finland. The campaigns study QoE and WTP related to Dynamic Adaptive Streaming over HTTP (DASH). They also confirm or disprove previous studies, openly share the data, and provide empirical background information on the purchasing behavior of customers. Due to the testing at two locations, we can further first time study whether cultural or regional differences affect the purchasing behaviors of such services. Additionally this paper gives insights and updated methodological guidance on conducting future WTP studies.**

I. Introduction

The success of online services depends on several factors such as the value they provide (to match and satisfy customer demand), Quality of Experience (QoE), pricing strategies, but also on optimal use of resources (for cost efficiency). While the value proposition of such services may primarily be defined by the provided contents (e.g., video content to be streamed; quality of the videos), communication services can substantially affect the experience for customers. From the networking research and business point of view, the questions of quality, pricing, resource management and the interplay of these factors are, thus, the most interesting ones in understanding roles and co-operation of operators, ISPs, end customers and other stakeholders.

Service providers and operators have certain trade-offs to take into account when dimensioning for their service. They can try and minimize their costs, risking a lower-quality service, or they can try and offer the best possible quality to their users, with the risk of being inefficient in terms of cost (as achieving high quality levels in online services most often involves a significant investment in terms of resources). Between those extremes, there is of course a range of cost / quality ratios that can be planned for. QoE research gives good indications on managing such kinds of trade-offs.

However, optimisations are not only possible on the dimensioning side, but are also necessary for pricing and market strategy: While not very common today, service providers can make use of price discrimination based on quality, customer segment, regional factors. etc. In the context of QoE, the quality-based discrimination where operators offer pricing tiers with correspondingly different service quality levels are of outmost interest. Doing so in an optimal manner requires an understanding of how users perceive the value of the service and service quality (i.e., its *utility*), and how it translates to monetary means (i.e., revenues). This is a significantly different assessment to classical QoE testings as service and quality appeal may not equally translate to purchases or (high) WTP.

In contemporary markets the service offering and pricing can face highly dynamic competition as new challengers try to enter the field or existing companies try to increase their market share. This may also affect the users' opinions and their expectations on quality and pricing. Therefore it is important to understand and be able to estimate what kind of effects (sometimes necessary) tariff changes may incur. Additionally, for a new service or company it is important to plan the market entry properly. Market entrance pricing is a key element in this planning. While low entrance pricing may attract users, the later increases may prove to be difficult to implement, renderering the business unsustainable.

In this paper we propose to address the question of how users perceive the value of better quality in an online video service (*à la* Netflix), by means of an experiment on their Willingness-To-Pay (WTP) for different quality levels, as a close metric to utilities for ISPs. We further investigate how the relation of QoE and WTP is affected by different tariff changes and cultural or regional effects. The work presented herein replicates and expands upon a previous work in [1], where the problem was systematically studied. The present work differs from the previous approach by using an entirely paperless test laboratory, but also recent codec, i.e., H.265 / HEVC, and video adaptation advancements, DASH. The present work can also be considered to be a retesting of the results in [1], with target of the trial data to be openly accessible for the research community, which is not the case with the previous results. The experiments were carried out with almost identical setups in two labs, at the University of Vienna, in Austria, and at VTT, in Oulu, Finland. This allows the unique comparison of regional effects that may affect the utility and, thus, WTP for

ISBN 978-3-901882-83-8

network video quality services.

From the results of the earlier work [1] we can isolate the following null hypotheses that were studied in this work:

Hypothesis 1 *WTP for network video quality upgrades does not exist.*

Hypothesis 2 *Historic pricing does not affect the market entrance of quality enhanced network video services.*

Hypothesis 3 *Different consumer segments do not make different quality - price decisions.*

The execution of the similar campaigns in two countries allows for testing possible variation in WTP between the two cultures. Additionally, it is known that the consumer prices (in relation to purchasing power) in Finland are higher than those in Austria (Comparative price levels 122.3 and 105.8, respectively[1]). Therefore we postulate the following null hypothesis:

Hypothesis 4 *WTP for quality-differentiated network video services is not affected by regional or cultural factors.*

The remainder of this work is structured as follows: in Section II we cover the relevant related work. The experiment environment, design and both setups are described in Section III before presenting the results in Section IV and analysis of key findings in Section V. The paper is finished with conclusions in Section VI.

II. Related Work

QoE has been a vital research topic in telecommunications for years. Especially the empirical perspective, both laboratory and field, to map the technical QoS to a subjective representation of QoE has received substantial attention [2], [3]. Numerous standards and recommendations, e.g., [4] and [5], have improved the test practices in order to obtain reproducible, consistent results. The transfer of empirical or estimated QoE data to the provisioning of network resources has been, for example, discussed in [6] and more access-oriented in [7]. Despite the usefulness of such data and practices, the economic utilization has been hampered by several knowledge gaps:

1) The mapping of QoE to purchasing or spending behaviours
2) Communication problems [8] due to the experience product nature [9] of network quality
3) Difficult generalisation of data across individual measurements [10]

The most pressing issue is the first one listed above, as QoE information needs to be transferred to perspective of business models: utilities, product demand, etc. While an early work [11] has targeted the assessment of WTP for network quality the community has been silent for years afterwards. In the last few years this problem was then finally targeted from several perspectives:

1) The fixed-point model of QoE [12] which formalises the interaction between price and subjective quality experience
2) Empirical confirmation of early WTP results in [13] and [1], as well as the exploration of QoE spending phenomena, e.g., related to cognitive dissonance [14]
3) Approximation of WTP from QoE and other results in [10]

The recent empirical efforts for understanding WTP have focused on careful laboratory setups by learning from the experiences in QoE testing. Contrary to the approach in [11], [13] and especially [1] have strictly moderated the information that is provided to the user. In other words, these studies have reduced the usage complexity and eliminated several biases, such as an inherent convergence to the mean (of the quality range) effect. While [13] has tested UDP video transmissions under packet loss, [1] has used more modern adaptive streaming technologies based on TCP. Both studies were able to illustrate a reasonable WTP for enhanced network services, a clear trade-off management of subjects between quality and price concerns, and effects induced by historic pricing (i.e., "market entrance pricing" recommendations). Despite the promising results, the results of these studies are not openly available and due to the low sample sizes a confirmation of the effects is advisable. This work will, hence, bring the test design used in [1] to 2015 by conducting a new campaign using up-to-date codecs and video adaptation techniques.

III. Experiments Description

A. Overview

The tested scenario was about watching typical video streaming content in a living-room like environment and making video quality purchasing decisions. In addition to the hypotheses to be verified or disproved by the empirical laboratory-based studies, the work had also some technical and generic goals. Recent developments in multimedia technologies called for considering them also in WTP studies (in addition to numerous QoE studies covering them). The technological advancements compared to previous studies are summarized in Section III-B.

B. Technological Advancements & Changes

While the technical setup followed the initial testing in [1], a series of changes and advancements were necessary in order to meet the state-of-the-art of technologies and to respond to insights from the earlier tests.

In a way analogous to the initial testing, but contrary to the older studies such as [11] and [13], our study used adaptive streaming over TCP to allow dynamically applying quality changes and also to match the contemporary typical video usage. The standard DASH [15] was used as video streaming technology, instead of Apple's HTTP Live Streaming (used in [1]). The DASH content was played out for viewing with the

[1]Eurostat, Purchasing Power Parities: http://ec.europa.eu/eurostat/web/purchasing-power-parities/ last accessed: March 30, 2016

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

GPAC client[2] for Linux. The GPAC player was manipulated in order to reduce the buffering and associated quality switching times, which was important for the experimental setup.

In earlier tests, the H.264 [16] video coding format was used. In the described campaigns, the substantially improved H.265 [17] (also referred to as HEVC [18], [19]) encoding was used instead. In pilot tests (executed prior to actual user campaigns), a reduced bandwidth demand of approximately 30% was witnessed in order to obtain comparable QoE values.

Contrary to a separate monitor in the initial testing, an iPad tablet computer was used to display both the available content (video library) and the price of the current selection (plus the information about the remaining reward of the user).

Finally, while in the initial testing 3 test groups were used, the test group design was simplified in our approach, as sketched below.

C. Experiment Environment and Contents

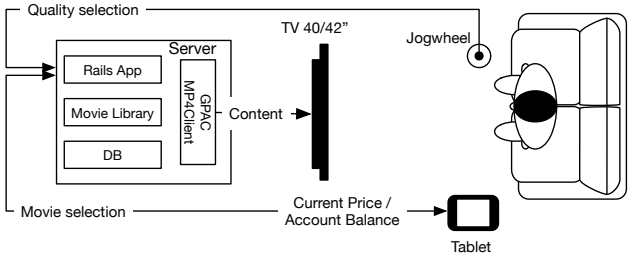


Fig. 1. Experiment set-up

1) Testing Environment: The subjective tests were executed in laboratory environment adhering the ITU-T P.910[4] (e.g., sample size and lighting conditions) and ITU-R BT.710 [5] (e.g., viewing distance) as closely as possible. The experiment set-up and how test subjects viewed and controlled the testing application is illustrated in Fig. 1. The main components of the test environment are described in Table I.

2) Tested Contents: The contents were prepared by extracting the content from the purchased Blueray discs into full quality versions (in M4V and Matroska containers). Then the 20 minute clips of each movie were carefully selected and edited in full quality. Finally, the full quality clips were transcoded to different degraded qualities with help of *x265*[3] and *FFmpeg*[4] tools. The actual qualities (defined in terms of bitrate) are specified in upcoming sections. The original contents available in the Movie Library of the campaigns are listed in Table II. Some videos were offered in English and German in the Vienna trial.

[2]GPAC Multimedia Open Source Project: http://gpac.wp.mines-telecom.fr/, last accessed: March 30, 2016

[3]http://x265.org/, last accessed: March 30, 2016

[4]https://www.ffmpeg.org/, last accessed: March 30, 2016

TABLE I
TEST ENVIRONMENT: MAIN COMPONENTS

Component	Description	Function
Server	Ubuntu 14.04 LTS PC	Host for the software application components.
Rails App	Rails application	Control UI for tablet and control logic for test.
Movie Library	File system	DASH video files in all quality levels.
DB	SQLite3 Database	Metadata and results of the tests.
GPAC MP4Client	Media player	Presentation of the DASH contents to the subjects.
TV 40/42"	TV set	The screen for viewing the contents.
Jogwheel	Jogwheel device	Remote control device for the video quality selection.
Tablet	iPad	Device for the selection of videos, interactive questionnaires, and the presentation of instructions, price information and deposit balance for subjects.

TABLE II
THE CONTENT AVAILABLE IN THE MOVIE LIBRARY

Content	Description	Campaign
Grand Budapest Hotel	Comedy, 2014.	Vienna (outliers)
Breaking Bad	TV series / Crime, 2012.	Both
The Dark Knight Rises	Superhero, 2012.	Both
Edge of Tomorrow	Science fiction, 2014.	Both
Guardians of the Galaxy	Superhero, 2014.	Both
Harry Potter and the Order of the Phoenix	Fantasy, 2007.	Both
Inception	Sci-Fi, 2010.	Both
Interstellar	Sci-Fi, 2014.	Both
Oblivion	Sci-Fi 2013.	Both
Oblivion	Sci-Fi, 2013.	Both
Orphan Black	TV series / Sci-Fi, 2013	Both
The Hobbit: An Unexpected Journey	Fantasy, 2012	Both
Transcendence	Sci-Fi, 2014	Both (different edits)
Toy Story	3D animation, 2010	Oulu

D. Test Design

During the tests the users could choose from eight quality classes, where the quality of each class was controlled in terms of bitrate. The different quality levels are named $Q_0 \dots Q_7$, where Q_0 denotes the class with the lowest and Q_7 the class with the best quality.

The test design used three tariffs A, B and C with linear price curves from 0 to the resp. maximum prices p_{max} of €2, €3 and €4:

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

$$A := \{p_0 = 0, p_1 = 0.286, \ldots, p_6 = 1.714, p_7 = 2\} \quad ,$$
$$B := \{p_0 = 0, p_1 = 0.429, \ldots, p_6 = 2.571, p_7 = 3\} \quad ,$$
$$C := \{p_0 = 0, p_1 = 0.571, \ldots, p_6 = 3.429, p_7 = 4\} \quad .$$

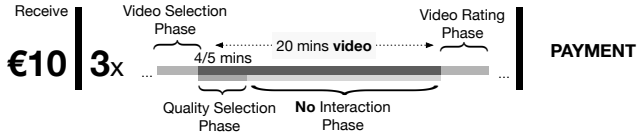


Fig. 2. Experiment sequence

The experiment process is illustrated in Fig. 2. First, each subject received €10 at the beginning of the trial – the money was shown as a deposit on the screen and was initially provided symbolically in cash. The subjects could (but needed not) use this money to finance quality upgrades during the trial (€10 were sufficient to constantly watch the best quality). After the trial, the remaining money on the deposit was to be paid out in cash, i.e., up to €10 could be paid out to the users in cash.

The actual experiment consisted of three measurements t_1, t_2, and t_3, each consisting of a 20 minutes video of the subject's choice and some ratings. Each user was assigned into *Group 1* or into *Control* group. The difference between the groups was in the tariffs (A, B, and C) that user was exposed to:

- *Group 1*: $t_1 : A \rightarrow t_2 : B \rightarrow t_3 : C$
- *Control*: $t_1 : B \rightarrow t_2 : B \rightarrow t_3 : C/A$

In other words, *Group 1* tested the *increasing* prices and the *Control* group had the stable pricing in t_1 and t_2 for comparison reasons. The *Control* group was further divided into two subgroups regarding the tariff of t_3 for a broader tariff comparison. Contrary to the initial trial, the decreasing prices were not tested, due to sample size reasons (in both the initial and retested trial) and the higher effects that have been witnessed for price increases in [1]. This test design allows within-subject comparisons, which require lower sample sizes for providing expressive results.

Analogously to the notions used in [13], each measurement t consisted of four phases (illustrated in Fig. 2):

1) **Video Selection Phase (VSP)**: The subject browsed our extensive library of modern video material and selected the content of her liking. The next phase was triggered upon the selection of the video.
2) **Quality Selection Phase (QSP)**: During the first 4 minutes (5 in Oulu trial) of the video watching the subject could freely test any quality level and evaluate the different quality-price tradeoffs (price was shown; quality was only perceived). When the QSP closed the quality level was fixed (to current one) and the price was finally deducted from the subjects balance.
3) **No Interaction Phase (NIP)**: The rest of the video was shown using the quality class selected by the user in the QSP. No further quality selection interaction was possible.
4) **Video Rating Phase (VRP)**: After the video had finished, the subject rated the QoE on the ACR-5 scale (*Bad*, *Poor*, *Fair*, *Good*, *Excellent*) and answered a binary acceptance question.

At the beginning, there was a pre-session questionnaire; the user was asked to specify her/his gender, age, education, Internet usage, Internet video purchasing habits and whether the user subscribes to some video services.

In a post-session questionnaire subject anwered the question "Did it feel like spending your own money?" in order to understand the validity of the test methodology. The users were also asked if they liked the available content.

E. Vienna Campaign

Using members of our faculty, several full-length pilot tests were conducted. Such tests served the elimination of test biases (such as unclear user interfaces), and assured the technical functioning of the trial and the meaningful parameterisation of the trial. Our expert users made the following noteworthy observations:

- The system is easy to use, the video content is interesting, and the scenario is realistic.
- Relative to H.264, H.265 performs surprisingly well with moderate bitrates
- Some experts did not recognise any quality gains above Q_4
- As the sound quality was rated to be insufficient, a new surround sound system was installed after the pilot test.

The actual campaign was conducted in our living laboratory in Vienna in July 2015. Twenty-two (22) test subjects completed all stages of the experiment with an average duration of ≈ 1.5 hours. Due to the three measurements, within-subject comparisons are enabled and 66 data points (purchases and associated QoE ratings) are available. 9 subjects ($\approx 41\%$) were female and 19 had graduated from a university (typically with a master's degree or equivalent). The subjects belonged to the following age groups: 2 subjects were between 10 and 19 years old, 11 were between 20 and 29, 6 were between 30 and 39 and 3 subjects were older. Their experiences with VoD services were limited, i.e., 11 subjects seldom purchased contents, one did so weekly, and 7 had one or more video service subscriptions.

F. Oulu Campaign: Differences to Vienna Campaign

Some technical improvements were implemented by the recommendations derived from Vienna trial:

1) One of the observations was that HEVC provides good quality already on rather low bitrates. While generally a positive development, the high efficiency of the codec decreased the width of active decision making area in the trial as most users reached acceptable quality levels

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

already on Q_3 or Q_4. To this end, the set of available quality levels were changed by adding more lower bit rate alternatives (See Table VI). The bit rates were selected so that their Peak Signal to Noise Ratio (PSNR) increases linearly. Also one of the videos was changed to start from beginning (Transcending) and one video was added to the library (Toy Story 3).

2) The price was made more prominent in control device by showing it in red colour.
3) The segment length of DASH content was set to 1 second (using also GOP size of 1 s). In the Vienna trial it was 2 seconds.
4) A 5 minute Video Selection Phase was used (as opposed to 4 minutes in the Vienna trials).

The test campaign was executed in November - December 2015 in the (living room-like) QoE laboratory at VTT premises in Oulu. Nineteen (19) subjects completed the experiment with average duration of $\approx$ 1 hour and 15 minutes. In this trial the three measurements led to 57 data points (purchases and QoE ratings). Most of the participants were VTT employees, and 4 of them were female ($\approx 21\%$) and 17 held a university degree. The subjects belong to the following age groups: 2 subjects were between 10 and 29 years old, 10 were between 30 and 39, 5 between 40 and 49 and two were older. They used Internet rather much as 8 persons reported over 5 hours daily usage and 10 reported the usage of 1 to 4 hours per day. 6 participants reported never buying video content from the Internet, while the remaining participants used to purchase content seldomly. 5 participants reported buying their video content in HD/4K quality. 11 participants subscribed to some video services, but only 2 persons subscribed to more than one service.

Due to the taxation laws of Finland, the reward could not be given out in cash (as intented). Instead the participants were rewarded with a movie ticket and variable amount of candy (depending on their deposit balance) they could select themselves. Nevertheless, the participants were led to believe in the beginning of the test that they would receive the €10 as in the original design. Please note that rewarding was not discussed during recruitment, so people were not expecting monetary compensation when they arrived to the tests.

IV. Results

A. Vienna Campaign Results

TABLE III
2015 trial (Vienna): Quality levels Q_0 to Q_{17} in kBit/s.

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
128	256	512	1024	2048	4096	8192	16384

In the Vienna trial, the video qualities shown in Table IV were tested for quality levels Q_0 (poorest) to Q_7 (best). For these values the WTP shown in Table IV was observed.

In other words, a substantial WTP for enhanced network video services was observed (median: €1.29 out of €2.98, the average maximum of the campaign), comparable to the experiences made in [1]. The majority of the subjects selected intermediary quality levels (see Fig. 5). Contrary to the trial reported in [1], no peaks towards the range extrema were observed. This may be explained by the chosen codec: using the modern H.265 codec, the QoE saturates quickly with the chosen bitrates—H.265 provides better than expected QoE improvement for low bitrates. The codec starts to perform very well already at moderate bitrates, which yields surprisingly high QoE ratings, as Mean Opinion Score (MOS) on ACR-5 scale — see Fig. 4. Hence, subjects may not have perceived any quality difference for qualities better than Q_4, which distributed the premium segment between Q_5 and Q_7. In a restesting, we recommended a finer-grained quality offering in the lower to medium QoS range.

TABLE IV
Spending per Tariff (*2015 trial, Vienna*).

	Overall	**Tariff A**	**Tariff B**	**Tariff C**
Max offering	€2.98	€2	€3	€4
Median	€1.29	€0.86	€1.71	€1.71
(% of maximum)	(43%)	(43%)	(57%)	(43%)
Std. deviation	€0.87	€0.43	€0.80	€1.16
(% of maximum)	(26%)	(22%)	(27%)	(29%)

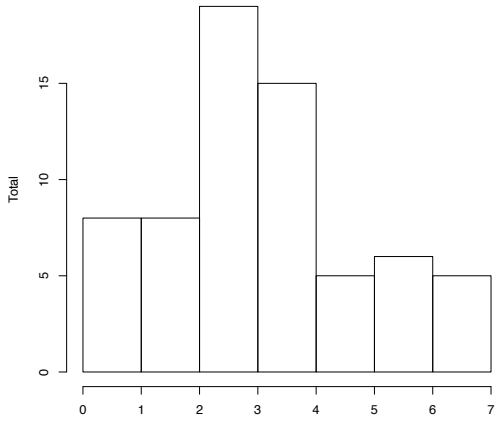


Fig. 3. Selected quality class Q_x (x-axis) in Vienna trial

The obtained WTP data was further relatively noisy — high variation in t_1, low correlation between t_1 and t_2 in the control group. This may indicate that the video files offered in the marketplace were too heterogenous to allow a direct comparison. Thus, we recommend a further improvement of the carefully selected video library to assure even higher consistency. As especially the first measurement t_1 was very noisy, we suggest a longer QSP duration – in the Vienna trial only 3 minutes were used in pre-trial testing, which was later on extended to 4 minutes for the actual trial.

Due to the noisy data, the detailed analysis of the used groups has not been conclusive. The high noise in t_1 affects the comparison of t_2 across groups (equal tariffing in t_2, but unequal historic tariffs). However, when focusing on t_2 and

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

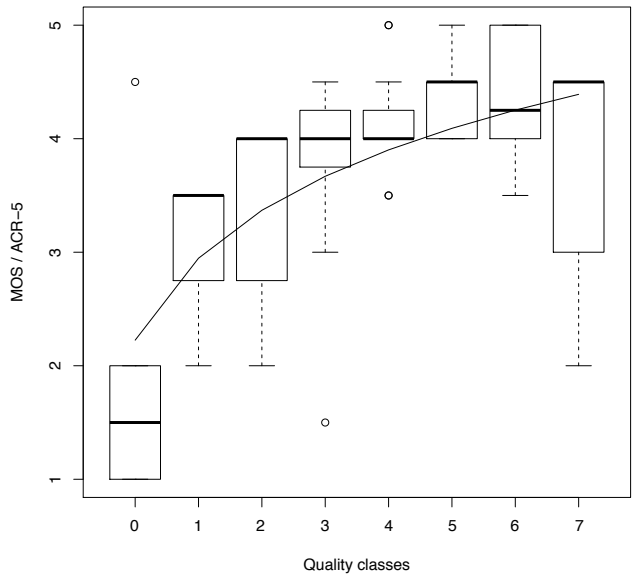


Fig. 4. Box plot of MOS ACR-5 ratings across all tariffs with logarithmic fit (*Vienna*).

t_3 of the control group, the subgroups with tariff sequences $B \rightarrow A$ and $B \rightarrow C$ can be compared.

TABLE V
SPENDING AS PERCENT OF p_{max} IN t_2 AND t_3 PER CONTROL SUBGROUP IN VIENNA TRIAL.

	t_2	t_3
	B: $p_{max} = 3$	**A:** $p_{max} = 2$
Mean	43%	45%
Median	43%	43%
	B: $p_{max} = 3$	**C:** $p_{max} = 4$
Mean	51%	29%
Median	57%	43%

As shown in Tab. V, the normalised expenditure is substantially affected by price increases, while price drops are hardly felt. When applying an ANOVA RM ($\alpha = 0.05$) to both the absolute and normalized spending, no significant time, group and group-time effects can be observed, however. This is caused by the test design that focused on the comparison of results in t_2 rather than t_3 and across test groups rather than looking at subgroups of the control group. Due to the high noise, not explained by rationales of subjects or provided feedback, especially in t_1, the analysis of historic pricing effects from [1] cannot be repeated for $t_1 \rightarrow t_2$.

An improved retesting shall target the working out of such effects in t_1 and t_2 or in a redesigned later test phase. The latter results highly correlate to the observations in [1]. Subjects seem to avoid a redecision in the case of price decreases, as not absolutely necessary, but immediately respond to price increases — see relationship to cognitive dissonance in [14].

Almost all subjects liked the provided contents (91%) and rated the "own money" feeling with 2.9 (ACR-5) on average (median: 3.0; "Fairly"). These results support the general functioning of the campaign design.

B. Oulu Campaign Results

TABLE VI
2015 TRIAL (OULU): QUALITY LEVELS Q_0 TO Q_{17} IN KBIT/S.

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
128	180	280	440	800	2548	8000	15000

The video qualities shown in Table VI were tested in Oulu trial. The overall and per tariff WTP are shown in Table VII. Interestingly, the WTP is higher than in the Vienna trial (€1.71). The results of both trials disprove Hypothesis 1.

As illustrated in Fig. 5 most of the users again selected intermediary quality levels. The peaks at the range extrema observed in the trial reported in [1] are present (esp. on the high-quality end), but they are not very pronounced. Still, the user groups present in [13] – (*price focused users*, *average users*, and *quality focused users* – can also be spotted in Fig. 5, which disproves the Hypothesis 3.

Fig. 6 illustrates the MOS of each quality level and it demonstrates the logarithmic nature of QoE. The implemented changes in the Oulu trial over the Vienna trial have caused more quality differentiation (by users) in medium and high bit rates, as was intented. This has likely affected also the selected qualities of Fig. 5.

TABLE VII
SPENDING PER TARIFF (*2015 trial, Oulu*).

	Overall	**Tariff A**	**Tariff B**	**Tariff C**
Maximum	€3.02	€2	€3	€4
Median	€1.71	€1.43	€1.71	€2.29
(% of maximum)	(57%)	(71%)	(57%)	(57%)
Std. deviation	€0.87	€0.43	€0.80	€1.16
(% of maximum)	(25%)	(18%)	(25%)	(27%)

Next, the campaign-wide mean expenditures and mean (selected) qualities including all the measurements (t_1, t_2, t_3) done by the users of the *Group I* and *Control* group were calculated. The mean expenditures of *Group I* and *Control* groups were €2.12 and €1.65, respectively, while mean qualities were 5.97 and 4.93, respectively. Both differences were tested with t-tests and found significant on alpha level 0.05 (p-value of 0.03 for expenditure and 0.02 for quality). Similar differences can be found in Vienna trial outcome, but less significant (p-value of 0.14 for both expenditure and quality). Also, when comparing t_2 expenditure of Control group is lower than expenditure Group I (€1.52 vs €2.06) with close

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

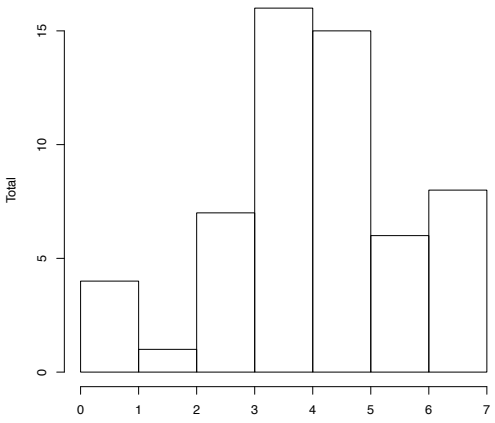


Fig. 5. Selected quality class Q_x (x-axis) in Oulu trial

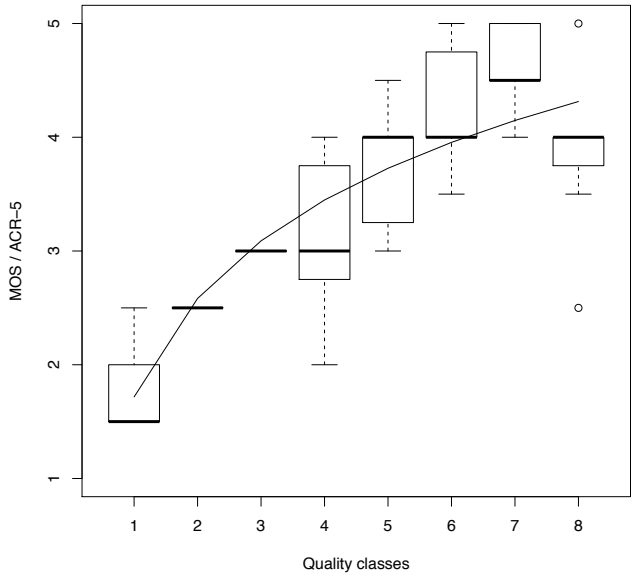


Fig. 6. Box plot of MOS ACR-5 ratings across all tariffs with logarithmic fit (*Oulu*).

to significance p-value (0.11). It has to be noted that with such a small sample size even a single user can affect the result.

Both of these observations contribute to disproving Hypothesis 2 (as the different pricing histories seem to affect), while we still cannot claim it to not hold.

Again focusing on t_2 and t_3 of the control group (c.f. Table V for Vienna results), the subgroups with tariff sequences $B \rightarrow A$ and $B \rightarrow C$ are compared. As shown in Tab. VIII, the results contradict those of the Vienna trial. In the Oulu trial the price drop has triggered redecision (or the users are not making an active decision, but follow an earlier price decision), while the price increases are hardly felt. Both trials show (weak) signal of price history affecting the price decision thus partially disproving Hypothesis 2.

Similarly to what happened in the Vienna trial, almost all users liked the content they chose (91 %). The subjects reported slightly higher "own money" feeling with 3.2 (ACR-5) on average (median: 3.5; "Fairly") than in Vienna trial.

TABLE VIII
SPENDING AS PERCENT OF p_{max} IN t_2 AND t_3 PER CONTROL SUBGROUP IN OULU TRIAL.

	t_2	t_3
	B: $p_{max} = 3$	**A:** $p_{max} = 2$
Mean	57%	79%
Median	64%	79%
	B: $p_{max} = 3$	**C:** $p_{max} = 4$
Mean	46%	43%
Median	57%	57%

V. ANALYSIS

A. Regional Differences in Expenditures

As described in Section IV, the realised expenditures of Oulu campaign were higher than ones in Vienna. It is possible that some cultural or socio-economic factor(s) affect the purchasing behaviour of participants. On the other hand, the difference could be explained by the altered quality levels between the campaigns.

The observed aggregated MOS of all measurements in both Vienna and Oulu campaign was 3.8, whereas the average selected quality class in Vienna was 4.5 and in Oulu 5.5. Also, it can be observed from Figure 4 and Figure 6, that the average QoE of 3.8 is reached at Q_4 in Vienna and at Q_5 in the Oulu trial. Therefore we can conclude that on the average, participants in Oulu chose quality level one step higher compared to the Vienna trial. Furthermore, the average spending in Vienna was €1.48 and in Oulu €1.90, their difference being €0.42. The average price increment in both campaigns is €0.43 which is very close to observed average spending difference. Also, the average bitrate in Vienna trial (4080 kbits/s) is 19 % larger than in Oulu trial (3422 kbits/s). The difference is very close to difference in WTP, 22 %. It can be concluded that the findings support the Hypothesis 4: the online video service market can be regarded to be of global nature where regional limitations may be marginal and relative to the cultural diversity (applicable at least between countries within economic and cultural proximity, but not necessarily between countries with highly different sosio-economic conditions).

B. Market Entrance and General Pricing

In the earlier work [1] it was observed that the price increases caused active decisions to be triggered (typically

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

leading to lower normalised expenditure), while price decreases did not trigger similar re-evaluations. The similar (yet weak) effects could be observed in Vienna trial, that suggest that aggressive entry pricing (discounts) may not be suitable for all markets.

On the other hand, in Oulu trial the effects seem to indicate the opposite (admittedly, weakly). There the subjects were more willing to tolerate the price increases, once they had originally made quality selection in the t_1/t_2. This implies that for some customer segments low entry price strategy (e.g. free/discounted first month) can be a viable option.

In the Oulu trial, when the prices were dropped for part of the *Control* group (c.f. tariff B in t_2 and tariff A in t_3 as shown in Table VIII) the normalised expenditure seems to increase. This could indicate that the participants had made the initial WTP decision already before and they were maintaining the expenditure level even after the price drop. Such customers could potentially allow introducing discount campaigns without losing much revenue while attracting new users (e.g. the existing users could get a quality upgrade for the same price).

Regarding the different set of available qualities and resulting WTP, we can observe that adding more low level quality steps has resulted in higher spending. One way to interpret this information, is to conclude that participants "had to" spend more to acquire the adequate quality (which subjects of both campaign seem to agree on). But in the end, customers spent more for the same quality in the latter trial. This highlights the importance of understanding the real willingness to pay for any offering (e.g. gained from market research prior to product launch) and optimising the pricing accordingly, so that all the potential revenue gets harvested.

C. Empirical Testability

Rather high variability in the results imply that the WTP testing differs drastically from traditional QoE testing. We can identify a few factors and recommendations that may have an effect on the WTP testability:

Active decision making: WTP study typically includes an active decision component (simulating the real-life purchasing event). Unlike in the act of perceiving a stimulus (e.g. watching a video), making a buying decision requires internal evaluation considering, for example, motives, the context and the potential value of the available object (better quality in this study). Presence of such evaluation makes the cognitive processing in WTP test very different than in typical QoE test setting.

Motivation heterogeneity: Varying motivations between subjects is likely to have an effect on QoE assessments as well as WTP assessments. Additionally, in a WTP study the motives of subjects may affect also the course of the test (via active decisions), which may not happen in more passive QoE tests.

Freedom: The consistency of the laboratory test results may benefit from the high level of control. However, for a WTP to be realistic people should feel free to do the buying decisions.

Perceived gain or loss: Subjective tests do not normally have a component of perceived loss or gain, but there is a static bilateral relationship between test conductor and test participant (contribution vs. reward). However, to be realistic WTP test, the *realistic* and *strong enough* gain-loss causality must be present (e.g. better quality, smaller reward). The subjects need to feel like spending their own money facing the "pain" component of purchasing event. This could be achieved e.g. by increasing the rewards and paid prices (if possible), or using innovative rewarding schemes (e.g. using chocolate or something concrete as a currency). Alternatively, if the test design allows, the "pain" component could be for example extra (and boring) task to be done.

Difficult parameterisation: Due to the high heterogeneity of motives, content preferences, and customer segments the parameterisation of this kind of campaign is generally difficult. Only when subjects are set in a critical situation where they have to actively manage the tradeoffs between quality and price, the exploration of motives or market entrance pricing effects is possible. Otherwise only the higher-level aggregate data can be obtained that gives a rough indication on the available demand and the associated WTP. In the Vienna trial, the high noise of the content appeared to be problematic.

The comparison of the Oulu and Vienna results shows that more inadequate low-quality offers, may lead to a higher revenue (while the quality choice may be similar). Partially this could be explained by the fact that in the Oulu trial, the price (an intentional test bias) may have played a too dominant role, i.e., the quality considerations have been secondary. This could explain the high relative adaptation under price losses and highlights the need to carefully moderate the required "price bias".

Market scenario: Having the participants in the right mindset, by the creation of a realistic market scenario within the trial, despite the limitations of empirical trials in general, is crucial to obtain the required data.

Assessment methodology: The WTP studies may also benefit from simplifying the assesment tasks. For example, binary choice offers (do you want to buy this quality level for this price?) may lead to different results than scenarios offering dozens of quality classes — see [10].

Content consistency: In case a test includes a variety of different contents (like in the described work), it is necessary to harmonize them regarding the studied properties. For example, the observed quality levels that guide the purchasing decisions (as in our test during the Quality Selection Phase), should be consistent across the contents and provide similar trade-offs to be considered by users.

VI. Conclusions

The results have shown that null Hypothesis 1 has to be rejected as substantial WTP was witnessed in both trials. Hypothesis 2 can be weakly rejected (due to lack of significance) as the historic pricing has triggered (at least weak) effects on purchasing behaviors. Interestingly, the effect of price changes varied between the trials. In Vienna trial, the

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

price increase caused normalised expenditure to drop, while in Oulu trial it was price decrease that triggered more significant effect, causing normalised expenditure to rise. Hypothesis 3 is rejected as there are different distinguishable customer segments (in Oulu trial), although not as clearly as in previous studies. Finally, the Hypothesis 4 is considered proved as the average WTP was on the same level in both trials (after compensating the effect caused by different quality levels between the campaigns). However, the hypothesis is proved only for regions and cultures with moderate sosio-economic differences (e.g. European welfare states).

Regarding the pricing aspects of video services, the results indicate that for some video streaming services the same pricing scheme can be applied successfully to different regional markets of the same geopolitical area. On a global market level, where cultures are targeted that do not moderately resemble each other culturally and economically, differences may still be observed, which cannot be answered quantitively from the conducted trials.

We also repeat the recommendation of earlier work, that companies need to be cautious about extremely low teaser discounts as increasing the prices later on can be challenging. On the other hand, for some customer segments/cultures the service/quality level lock-in may prove to be strong enough to allow later tariff increases (as indicated by Oulu results). Finally, the results imply that clever pricing and packaging of the same product (additional lower quality levels in this work) can potentially increase the profits in some cases.

The trials have shown the challenging nature of conducting WTP trials (compared to traditional QoE testing). Some challenges were identified and recommendations are given in Section V-C. The future retestings shall take these further recommendations into account in test design.

VII. Acknowledgments

Martín Varela's and Toni Mäki's work was partially funded by Tekes, the Finnish agency for research innovation, in the context of the CELTIC+ project NOTTS. The research leading to these results has partially also received funding from the European Community's Seventh Framework Programme for the PRECIOUS project under grant agreement no. 611366.

References

[1] P. Zwickl, A. Sackl, and P. Reichl, "Market Entrance, User Interaction and Willingness-to-Pay: Exploring Fundamentals of QoE-based Charging for VoD Services," in *Proc. of the IEEE Globecom'13*, 2013, pp. 1310–1316.

[2] N. Staelens, P. Coppens, N. Van Kets, G. Van Wallendaef, W. Van den Broeck, J. De Cock, and F. De Turek, "On the impact of video stalling and video quality in the case of camera switching during adaptive streaming of sports content," in *Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2015, pp. 1–6.

[3] C. Keimel, A. Redl, and K. Diepold, "The TUM High Definition Video Datasets," in *Proc. of the Fourth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2012, pp. 97–102.

[4] International Telecommunication Union, "Subjective video quality assessment methods for multimedia applications," *ITU-T Recommendation P.910*, April 2008.

[5] *ITU-R BT.710-4 Subjective assessment methods for image quality in high-definition television*, ITU-R BT.710-4, ITU, 1998.

[6] F. Agboma and A. Liotta, "Qoe-aware qos management," in *Proc. of the Sixth International Conference on Advances in Mobile Computing and Multimedia*. ACM, 2008, pp. 111–116.

[7] K. Ivešić, L. Skorin-Kapov, and M. Matijašević, "Cross-layer QoE-driven Admission Control and Resource Allocation for Adaptive Multimedia Services in LTE," *Journal of Network and Computer Applications*, 2014.

[8] M. Varela, P. Zwickl, P. Reichl, M. Xie, and H. Schulzrinne, "Experience Level Agreements (ELA): The Challenges of Selling QoE to the User," in *Proc. of the ICC 2015 Workshops*. IEEE, 2015, pp. 1741–1746.

[9] P. Nelson, "Information and Consumer Behavior," *Journal of Political Economy*, vol. 78, no. 2, pp. 311–329, 1970.

[10] P. Zwickl, P. Reichl, L. Skorin-Kapov, O. Dobrijevic, and A. Sackl, "On the Approximation of ISP and User Utilities from Quality of Experience ," in *Proc. of the Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2015.

[11] FP5 Project M3I, IST–1999–11429, "Deliverable 15/2 – M3I user experiment results," 2002.

[12] P. Reichl, P. Maillé, P. Zwickl, and A. Sackl, "A Fixed-Point Model for QoE-based Charging," in *Proc. of the ACM SIGCOMM Workshop on Future human-Centric Multimedia Networking (FhMN)*, 2013, pp. 33–38.

[13] A. Sackl, S. Egger, P. Zwickl, and P. Reichl, "The QoE Alchemy: Turning Quality into Money. Experiences with a Refined Methodology for the Evaluation of Willingness-to-pay for Service Quality," in *Proc. of the Fourth International QoMEX Workshop*. IEEE, 2012, pp. 170–175.

[14] A. Sackl, P. Zwickl, S. Egger, and P. Reichl, "The Role of Cognitive Dissonance for QoE Evaluation of Multimedia Services," in *Proc. of the 2012 IEEE Globecom Workshops*. IEEE, 2012, pp. 1352–1356.

[15] *ISO/IEC 23009-1:2014: Information Technology – Dynamic Adaptive Streaming over HTTP (DASH) – Part 1: Media Presentation Description and Segment Formats*, ISO/IEC 23009-1:2014, International Standards Organization (ISO), 2012.

[16] *H.264: Advanced video coding for generic audiovisual services*, Recommendation H.264 (02/14) (twinned), International Telecommunication Union (ITU), 2014.

[17] *H.265: High efficiency video coding*, ITU-T H.265 (V3) (04/2015 (twinned), International Telecommunication Union (ITU), 2015.

[18] *ISO/IEC 23008-2:2015: Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding*, ISO/IEC 23008-2:2015 (twinned), International Organization for Standardization, 2012.

[19] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

Economics Models and Policies for Cloud Federations

George Darzanos, Iordanis Koutsopoulos, George D. Stamoulis
Athens University of Economics and Business, AUEB, Athens, Greece
{ntarzanos, jordan, gstamoul}@aueb.gr

***Abstract*—Cloud federation has emerged as an effective solution offering worldwide coverage, dynamic infrastructure scaling and improved QoS for the demanding cloud services. In this paper, we present a model for Cloud Service Providers (CSPs) federation and we investigate the economic benefits of CSPs under different federation modes. Each CSP is modeled as an M/M/1 queue with arrivals corresponding to computation tasks and service rate that captures the computational capabilities of the CSP. Each CSP earns revenue by charging its customers according to a QoS-dependent pricing function, and it undergoes a cost due to energy consumption of its infrastructure. We propose a model for the formation of cloud federations, according to which each CSP may forward part of the workload stemming from its customers to other CSPs. We define three federation modes with varying degrees of CSPs' interaction, namely the strong, weak and elastic federations. In strong federations, the CSPs jointly decide on their forwarding policies to maximize the total profit of the federation; then, they share these profits according to certain profit sharing policies. In weak federations, a game arises, in which each CSP follows a forwarding policy that aims to maximize its individual payoff, which however incorporates some fairness. In elastic federations, each CSP again aims to maximize its individual payoff, but it has the freedom to tune the degree of its selfishness through a pricing function. The numerical results validate and quantify the conjecture that federation can incur substantial monetary benefits and achieve a near to optimal QoS.**

I. Introduction

Nowadays, the multi-faceted applications that are moving to the cloud demands global geographic presence and high QoS for end-users. Although Cloud Service Providers (CSPs) promise flexible and scalable resources, thus creating the illusion of infinite resources to their customers, no CSP can provide on-demand dynamic resource scaling in order to handle the workload variations in a cost-effective manner. Furthermore, even market giants have limited geographic coverage since it is not profitable to invest on establishing datacenters in multiple geographical locations to satisfy the demand. Cloud federation arises as an effective way to expand the reach of CSPs and improve the QoS of their customers.

In a cloud federation, multiple CSPs cooperate to provide seamless provisioning of high-quality services across different domains. A cloud federation should be accompanied by certain policies that ensure the sustainability of this CSP community, and each CSP that participates has to conform to these policies. The policies should guarantee that each CSP that joins the federation will not undergo profit loss. Further, they need to motivate all CSPs to participate regardless of their market power or the size of their infrastructure. Cloud federation comes together with several participation incentives such as geographic footprint expansion, the scaling of resources to handle the request traffic bursts of peak demand and the inter-cloud load balancing. Hence, a cloud federation prevents the datacenter over-dimensioning and further it reduces the CSPs' energy cost through better utilization of their infrastructure.

In real life, there are several instances of cloud federation in both academic and enterprise environments. The OnApp Federation [1] is a network of CSPs running on the OnApp cloud management platform. The CSPs that join this federation may buy and sell capacity on demand through the OnApp market. Arjuna's Agility framework [2] is a dynamic federated cloud computing platform that is created from IT resources that are offered by autonomous, cooperating business parties within and beyond an enterprise, and under certain policies. EGI Federated Cloud [3] is a seamless grid of academic private clouds and virtualized resources, built around open standards and it focuses on the requirements of scientific community. BonFIRE [4] offers a federated testbed that supports large-scale testing of applications, services and systems over multiple, geographically distributed, heterogeneous cloud and network testbeds. Finally, the CERN Openlab project [5] aims to build a seamless federation among multiple private and public cloud platforms on OpenStack.

Several works in recent literature investigate the problem of resource allocation in cloud federations. These works can be classified into two broad categories: *(i) Cooperative resource pooling* [6]–[8], where CSPs aggregate their resources aiming to maximize the total utility of federation and *(ii) Resource trading* [9], [10], where CSPs aim to maximize their individual profit by trading their unused resources. In our prelude work [11], we modeled each CSP as an M/M/1 queueing system and devised a mathematical model for the net profit of each CSP. This consists of accrued revenues from pricing on its customers and of incurred energy cost at the cloud infrastructure. Furthermore, we introduced a first approach to model a service-oriented cloud federation between two CSP, where each CSP may forward a portion of the tasks stemming from its customers to other CSPs. Finally, we formulated the problem of finding the utility-optimal federation as a global profit maximization problem in which CSPs align their strategies to jointly solve it.

In this paper, we build on and substantially extend our previous work by studying different cloud federation regimes. In particular, we define the strong, weak and elastic federation

ISBN 978-3-901882-83-8

modes. Each mode differs on the level of cooperation among CSPs, the extent of private information that CSPs should make available to others, and the CSPs' objectives that may be aligned or conflicting. *Strong* federation requires an offline mutual commitment of CSPs, such that they all agree to align their forwarding policies to optimize their total net profit. Additionally, a mutually agreed policy is applied for the fair sharing of total profit of federation. In *weak* federation, the CSPs are still able to forward tasks, however each of them acts unilaterally by trying to maximize its own net profit, and thus an non-cooperative game arises. Nevertheless, the net profit of each CSP in this type of federation is strongly connected to its contribution in the federation, namely its profit share is given by its Shapley value. It will be seen that use of Shapley value as payoff function leads the relevant game to an efficient equilibrium. We also develop a more *elastic* model for cloud federations whereby all CSPs employ a flexible pricing scheme on forwarded tasks that reflects their degree of selfishness. Again, each of them aims to maximize its net profit and thus a non-cooperative game arises, the outcome of which depends on this degree of selfishness. For each federation mode we formulate the problem of net profit-optimal service delegation and we find the optimal forwarding policies.

The paper is organized as follows. In section II we provide an overview of relevant state-of-the art work. In section III, we present our model for a single CSP. In section IV we present our cloud federation model, we introduce and specify the three modes of federation and solve the relevant optimization problems. In section V we present our numerical evaluation, and in section VI we briefly present our conclusions.

II. Related Work

Architectural approaches of cloud federation. The authors in [12] present the challenges of a utility-oriented cloud federation and propose three basic entities for a market-based cloud federation architecture; the cloud exchange as the entity that creates the market, a cloud coordinator per CSP as seller and a cloud broker per client as buyer. The Reservoir model, a modular cloud architecture, is proposed in [13]. In Reservoir, multiple CSPs collaborate in order to create a virtual pool of resources that seems infinite. The authors in [14] present the concept of cloud federation as service aggregation and they present two modes of such a federation, the redundancy and migration federations. In redundancy federation, multiple CSPs come together and jointly offer a service to achieve improved quality for a client, while in migration federation a client is moved from an old service to new one offered by another CSP due to improved quality. Finally, the authors in [15] envision the federation of CSPs as vertical stack that fits on the layered model of cloud computing. A service request may arrive in any layer of a CSP and can served either by local resources using delegation to a lower layer or by another federated CSP using delegation to a matching layer.

Cooperative inter-cloud resource allocation. The authors in [16] propose cooperative price-based resource allocation mechanisms in dynamic cloud federation platforms, aiming to maximize the total utility of federation. In [6] and [7], coalitional game theory is applied as a mechanism for the dynamic formation of CSPs' federation. Both these papers have proposed algorithms that determine the optimal coalitions for a set of CSPs, given their client generated workloads. In [8] the inter-CSP VM migration is presented a solution to the problem of resource over-provisioning. The authors propose a global scheduler that decides whether a VM should migrate or shut down, thus aiming to CSPs utility maximization.

Resource allocation among selfish CSPs. In [9], the federation among geo-distributed CSPs is investigated. The authors design double-auction based algorithms for inter-cloud VM trading in federations of selfish CSPs. The authors in [17] model each CSP as a set of heterogeneous servers, each of them modeled as a queueing system. Then, they formulate the problem of resource allocation in a multi-CSP environment as a game among selfish CSPs, where each CSP aims to maximize its individual utility taking into account the customer SLAs. The author in [10] investigates the interactions among CSPs as a repeated game among selfish players that aim at maximizing their profit by selling their unused resources in a spot market. The model incorporates information for both historical and expected future revenue as part of the resource trading decision, in order to simultaneously maximize the CSP revenue and avoid future workload fluctuations.

Some of the above works provide an overview of the architectural elements of a federated system, while others consider the problem of resource allocation in inter-cloud environments of either cooperative or selfish CSPs. In our work, we propose the concept of federation among CSPs as service delegation and we model the federated environment as well as its involved economics. Contrary to most existing works, we provide policies both for cooperative and the non-cooperative federated environments. Further, we propose a flexible policy where CSPs can move between cooperation and selfishness. Additionally, most of existing works do not take into account the QoS offered to CSPs' customers in their optimization approach. In our work, the federation policies are optimal with respect to total or individual CSPs' profit (depends on policy), but they are also beneficial with respect to the QoS offered to customers.

III. Cloud Service Provider Model

A. CSP as an M/M/1 Queueing System

For each CSP i, we use C_i to denote the total computational capacity (in operations/sec) of its infrastructure. We assume that tasks from its customers arrive to a central controller according to a Poisson process of rate λ_i (tasks/sec). Each of these tasks requires a random number of operations in order to be executed. We assume that the number of operations follows an exponential distribution with mean number L operations/task. The average service rate (in tasks/sec) for a CSP i is $\mu_i = C_i/L$, and thus the service time of a task is exponentially distributed with mean $1/\mu_i$.

We use the *average task completion time* as a metric for customers' QoS. By standard theory for M/M/1 single-server

queueing systems, the average completion time d_i for tasks served by the infrastructure of CSP i is given by

$$d_i(\lambda_i) = \frac{1}{\mu_i - \lambda_i}. \quad (1)$$

The average rate of incoming tasks must always be lower than the service rate of the system ($\lambda_i < \mu_i$), otherwise the CSP queue becomes unstable.

M/M/1 abstraction justification. In practice, a CSP consists of multiple datacenters with servers within each of them. In our approach, we abstract the multi-server infrastructure of the CSP as a single-server M/M/1 queueing system. To this end, we assume that a CSP performs perfect dispatching and scheduling of incoming tasks by preventing its servers from becoming idle. In particular, if the infrastructure of a CSP i consists of M_i identical servers of computational capacity C_i/M_i each, the CSP achieves the same average utilization level ρ_i to all servers by applying the optimal internal task dispatching and scheduling. Hence, we can safely assume that the multi-server infrastructure of each CSP behaves as a *single-server* with computational capacity C_i and utilization ρ_i.

The queueing-system assumption can be justified as follows: Tasks arrive at the controller in the form of a stream. Since cloud computing provides the technology for virtualizing resources, tasks coming from the customers of a CSP are assigned to established virtual machines (VMs). VMs are not in abundance, but they are finite resources that are assigned on-demand to serve requests. Given that a typical cloud computing system serves a large number of customers where each of them generates multiple computational tasks and these arrive in bursts, is more probable to have smaller interarrival times than larger ones. Thereafter, we can assume that the tasks arrive according to a Poisson process. Furthermore, the time that a task spends in the CSP's system depends both on the waiting and service time, i.e. on the number of existing tasks that wait to be served, on the availability of resources when the task arrives and on its size with respect to the number of operations it entails. The majority of tasks that arrive in a CSP queue usually demands a smaller number of operation, while relatively fewer tasks require a large number of operation. Hence, we can assume that the number of operations that a task requires is exponentially distributed, and therefore the service time also follows an exponential distribution. Consequently, the M/M/1 queueing model is applicable. While this is a simplification that allows the mathematical treatment of our paper, it is also reasonable enough to capture the reality.

B. CSP Economics

Energy consumption cost. We take the infrastructure energy consumption cost of a CSP as measure for its total cost. The power consumption of a server includes the power for its operation and the power that is required for supportive systems like cooling devices. However, according to prevalent state-of-the art literature [18], the total power consumption is a linearly increasing function of the utilization factor of the server, ρ. Specifically, the total power consumed is the sum of server's idle power and utilization factor-dependent dynamic power consumption. The former amount of power, W_0, is the power consumed when the server is powered on but does not serve any task. The latter one is linearly increasing in the server utilization ρ. If we denote by W_1 the power of a server when it is fully utilized (namely at $\rho = 1$), the range of power consumption is $[0, W_1 - W_0]$.

To estimate the total power consumption of a CSP, we take into account that its infrastructure consists of multiple servers. Since a CSP achieves the same average level of utilization ρ in all its servers (subsection III-A), idle and dynamic power consumptions of the entire infrastructure can be computed by aggregating the corresponding power consumption patterns of all servers. Consequently, if a CSP i has M_i servers, and if $W_{0,ij}$ and $W_{1,ij}$ denote the idle and total power consumption of the j-th server of CSP i, the aggregate power consumption of the CSP i in Watts is

$$\begin{aligned} W_i(\lambda_i) &= \sum_{j=1}^{M_i} W_{0,ij} + \frac{\lambda_i}{\mu_i} \sum_{j=1}^{M_i} \Big(W_{1,ij} - W_{0,ij} \Big) \\ &= W_{0,i} + \Big(W_{1,i} - W_{0,i} \Big) \frac{\lambda_i}{\mu_i}, \end{aligned} \quad (2)$$

where $W_{0,i}$ and $W_{1,i}$ denote the idle and total power consumptions of i's infrastructure. If i uses electricity at a price Z_i per KWatt·sec, the *cost of energy consumption* per unit of time is given by

$$E_i(\lambda_i) = W_i(\lambda_i)\ Z_i. \quad (3)$$

QoS-dependent Pricing. We assume that a CSP charges its customers based on the offered QoS-level and on load of received requests. Recall that we use average tasks completion time as measure for the QoS offered by a CSP. Thereafter, a CSP i sets a price per task according to a pricing function $p_i(\cdot)$, where $p_i(\cdot)$ is decreasing in average task completion time, d_i. This function should also be convex, because a marginal change in delay is perceived more by the customer for smaller values of the delay. Further, the average completion time of task is always lower-bounded by the expected service time $\beta_i = 1/\mu_i$. A function that satisfies the requirements above is

$$p_i(\lambda_i) = \frac{\beta_i}{d_i(\lambda_i)}\ Q_i, \quad (4)$$

where Q_i denotes the price per task that i charges for offering service in the best possible QoS, i.e. the expected service time β_i. In practice, the pricing function for each CSP is driven by the competition in the cloud market. In our approach, we assume that each CSP has made a decision offline on its pricing function that already takes into account this competition. Moreover, we assume that CSPs cannot adapt their pricing functions and also that their customers are committed by some contract and therefore they cannot change their serving CSP.

Revenue. The revenue of a CSP is generated from pricing on the tasks coming from its customers. Since CSP is also committed by some contract, we assume that the tasks arrive in its queue are always served. Consequently, the revenue rate in monetary units per unit of time for CSP i is given by

$$R_i(\lambda_i) = \lambda_i\ p_i(\lambda_i). \quad (5)$$

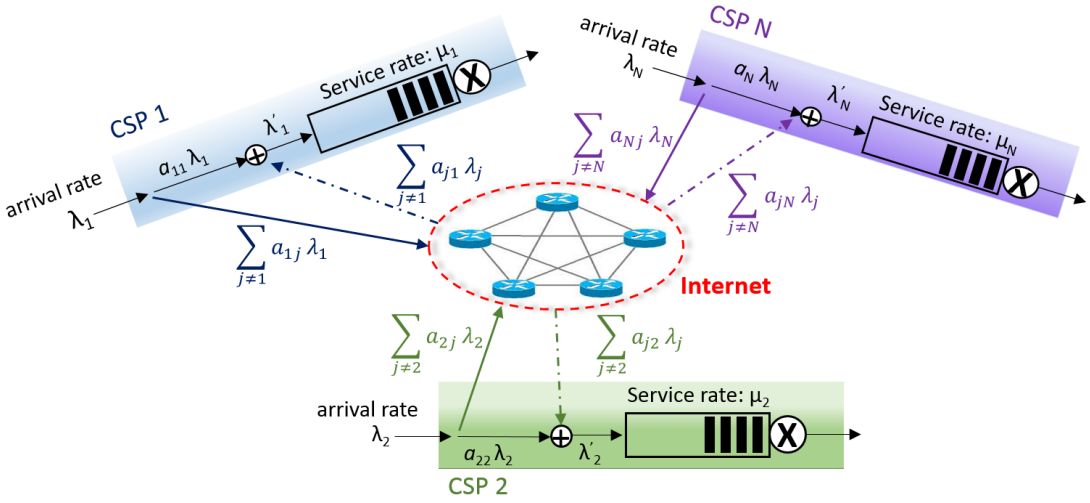


Fig. 1. Cloud federation for N CSPs, each of them is modeled as a single-server M/M/1 queue. Each CSP may forward a portion of the tasks coming from its customers to other CSPs and likewise it can receive task streams coming from customers of other CSPs. The streams of forwarded tasks between CSPs i and j undergo a fixed average transfer delay D_{ij}.

Net Profit. The net profit that i earns per time unit is

$$P_i(\lambda_i) = R_i(\lambda_i) - E_i(\lambda_i). \tag{6}$$

IV. Cloud Federation Policies

Our cloud federation model is based on the ability of each CSP to forward part of its incoming traffic stream of tasks to other CSPs within the federation. Therefore, the forwarding policy of each CSP is considered as its strategic leverage. The objective of a CSP for joining the federation may vary, and thus a CSP may have incentives to act either cooperatively or selfishly. We investigate three different modes under which the CSPs can federate:(i) the strong, (ii) weak and (iii) elastic federation modes. Each mode differs from others either in the level of private information that each CSP should make available to other CSPs or in the cooperation level of CSPs that may have common or conflicting federation objectives.

A. Model

We consider a set $\mathcal{N}$ of $N = |\mathcal{N}|$ CSPs, and for each CSP $i \in \mathcal{N}$ we define variables α_{ij} for $j = 1, .., N$ that determine the portion of its incoming tasks that CSP i forwards to a CSP j. Therefore, our global forwarding policy is a $N \times N$ dimensional matrix $\mathbf{A}$, whose entries α_{ij} determine the forwarding policy of all CSPs. We use vectors $\mathbf{a}_i$ and $\mathbf{a}'_i$ to denote the i-th row and i-th column of $\mathbf{A}$ respectively. The aggregate rate of tasks that CSP i forwards to others is $\sum_{j\in\mathcal{N}\setminus\{i\}} \alpha_{ij}\lambda_i$, while the average rate of tasks that CSP i receives from other CSPs is $\sum_{j\in\mathcal{N}\setminus\{i\}} \alpha_{ji}\lambda_j$.

Fig. 1 depicts our federation model for N CSPs. We assume that the portion of tasks that are transferred from a CSP to another, experiences an additional delay due to the intervening Internet links between their datacenters. Therefore, for each pair of CSPs $i, j \in \mathcal{N}$ we define an average communication delay D_{ij}. This delay is understandably exogenous to the system of CSPs. Also, we assume that the tasks that arrive in all CSPs belongs to the same service class and thus have the same mean number of operations, L, per task.

In our model, the task arrival rate at the input of each CSP's queue depends on the forwarding policy of other federated CSPs. Therefore, the ultimate arrival rate of tasks in the queue of CSP i depends on values of i-th column of matrix $\mathbf{A}$ (i.e. on vector $\mathbf{a}'_i$) and is defined as $\lambda'_i(\mathbf{a}'_i) = \sum_{j\in\mathcal{N}} \alpha_{ji}\lambda_j$. Thus, the average completion time of the tasks that are served by the infrastructure of CSP i is

$$d_i(\mathbf{a}'_i) = \frac{1}{\mu_i - \lambda'_i(\mathbf{a}'_i)}. \tag{7}$$

A portion of the task stream that arrives in a CSP is served by its own infrastructure, while other portions may be forwarded to other CSPs. Hence, the average completion time of tasks coming from the customers of CSP i depends on the average delay experienced at other CSP queues. Thus the average task completion time for customers of CSP i depends on all columns of matrix $\mathbf{A}$ and is defined as:

$$T_i(\mathbf{A}) = \sum_{j\in\mathcal{N}} \alpha_{ij}\big(d_j(\mathbf{a}'_i) + D_{ij}\big). \tag{8}$$

Note that $D_{ii} = 0$. At this point, it is important to stress the difference between $T_i(\cdot)$ and $d_i(\cdot)$:

- $d_i(\cdot)$ the average completion time for tasks that are served by i's infrastructure, including tasks originating from customers of i and tasks from other CSPs' customers.
- $T_i(\cdot)$ the average completion time of tasks that are generated from customers of CSP i, regardless of whether they are ultimately served by CSP i or by other CSPs.

In Section III a complete characterization of a single CSP is provided, however we need to slightly revise our model in order for it to be applicable in the federation. Now, the power consumption of i's infrastructure is affected by

forwarding policies of CSPs. Thus, the power consumption of i's infrastructure is given by

$$W_i(\mathbf{a}'_i) = W_{0,i} + \big(W_{1,i} - W_{0,i}\big)\frac{\lambda_i^{'}(\mathbf{a}'_i)}{\mu_i}. \quad (9)$$

Accordingly, the energy cost per unit of time is defined as

$$E_i(\mathbf{a}'_i) = W_i(\mathbf{a}'_i)\ Z_i. \quad (10)$$

The customers of CSP i should be charged based on $T_i(\cdot)$ rather than $d_i(\cdot)$ because different tasks may be served from different CSP queues. Hence, the pricing function becomes $p_i(\mathbf{A}) = \frac{\beta_i}{T_i(\mathbf{A})}Q_i$, and thus the revenue per unit of time is

$$R_i(\mathbf{A}) = \lambda_i \frac{\beta_i}{T_i(\mathbf{A})}Q_i. \quad (11)$$

Finally, the generated profit per unit of time is given by

$$P_i(\mathbf{A}) = R_i(\mathbf{A}) - E_i(\mathbf{a}'_i). \quad (12)$$

B. Strong Federation

All CSPs that participate in a strong federation comply to certain cooperation rules that have been agreed a priori. These rules include: (i) cooperation on exchanging private information, i.e. the values of their computational capacity C_i and average request load λ_i, (ii) agreement on the common objective of total federation profit maximization (iii) cooperation on defining the appropriate policy for sharing the total profit incurred from federation, and (iv) commitment to always serve the forwarded tasks of other federated CSPs.

Total Profit maximization. The CSPs cooperate and jointly decide the forwarding policies $\mathbf{A}$ that maximize the total federation profit. The globally optimal forwarding policy $\mathbf{A}^*$ is derived by solving the total profit maximization problem,

$$\begin{aligned} \arg\max_{\mathbf{A}} \quad & \sum_{i\in\mathcal{N}} P_i(\mathbf{A}) \\ \text{s.t.} \quad & \alpha_{ij} \geq 0 \quad , \quad \forall i,j \in \mathcal{N}, \\ & \sum_{j\in\mathcal{N}} \alpha_{ij} = 1 \quad , \quad \forall i \in \mathcal{N}, \\ & \lambda_i^{'}(\mathbf{a}'_i) < \mu_i \quad , \quad \forall i \in \mathcal{N}. \end{aligned} \quad (13)$$

The second constraint captures the splitting of CSP i's task traffic across others. The third constraint is due to stability in the queues of each CSP. We can solve this non-linear problem by applying standard optimization methods, i.e. formation of the Lagrangian and statement of the necessary and sufficient KKT conditions that should be satisfied for optimality.

Profit Sharing Policies. Our problem formulation guarantees that under the optimal $\mathbf{A}^*$, the total federation profit is maximized. Thereafter, in the worst case scenario, i.e. in $\mathbf{A}^* = \mathbf{I}$ (Identity matrix), the total profit of federation equals the aggregate profit of CSPs in standalone operation. By standalone, we mean that each CSP serves only the tasks coming from its customers. However the individual profit may in fact deteriorate for one (or more) CSPs due to task forwarding actions. Specifically, the CSPs that only receive forwarded tasks may have loss because the extra workload will increase their energy cost due to the higher infrastructure utilization. As a result, these CSPs may be unwilling to comply with the federation, unless some rule is applied for the elimination of their losses. Since the total profit of the federation exceeds the aggregate profit of CSPs in the standalone mode, CSPs that only forward tasks definitely have higher profit than before, thus they are able to compensate others. Therefore, the CSPs have to reach an agreement for the fair sharing of total generated profit that satisfies all of them.

Next, we present two cooperative profit-sharing policies that serve the objective above. In the first policy, the profit share that a CSP receives depends both on its standalone profit and on the percentage of total forwarded tasks that it forwards or receives. In the second policy, we determine the profit that a CSP should get based on its marginal contribution in the federation by making use of Shapley value notion [19].

1) Interaction driven profit-sharing: In this approach, a CSP i gets at least the profit it had in standalone operation, while the extra profit generated from the federated operation is proportionally shared among N CSPs based on the percentage of forwarded tasks that each of them forwarded or received. We define the *extra generated profit* $P_F(\mathbf{A}^*)$ by subtracting the aggregate profit of CSPs in the standalone operation from the total profit of federation

$$P_F(\mathbf{A}^*) = \sum_{i\in\mathcal{N}} P_i(\mathbf{A}^*) - \sum_{i\in\mathcal{N}} P_i(\mathbf{I}). \quad (14)$$

where $P_i(\mathbf{I})$ denotes the profit of CSP i in standalone operation. Consequently, the share of CPS i is determined by:

$$\xi_i(\mathbf{A}^*) = \frac{|\lambda'_i(\mathbf{a}'^*_i) - \lambda_i|}{\sum_{j\in\mathcal{N}} |\lambda'_j(\mathbf{a}'^*_j) - \lambda_j|} P_F(\mathbf{A}^*) + P_i(\mathbf{I}), \quad (15)$$

where $\frac{|\lambda'_i(\mathbf{a}'^*_i) - \lambda_i|}{\sum_{j\in\mathcal{N}} |\lambda'_j(\mathbf{a}'^*_j) - \lambda_j|}$ is the *proportionality* parameter which defines that a CSP who forwards or receives more tasks compared to another, will receive proportionally larger share of the extra generated profit.

2) Shapley value driven profit-sharing: Shapley value has been widely used in coalitional game theory applications as a mechanism for sharing total utility in a fair manner. A characteristic function $\upsilon(\cdot)$ measures the benefit of a coalition, also called the *worth of coalition*. In our approach, we take as characteristic function the total profit that is generated from the federated operation of a given set of CSPs. For instance, the worth of coalition $\upsilon(\cdot)$ for the set of $\mathcal{N}$ CSPs is

$$\upsilon(\mathcal{N}, \mathbf{A}) = \max_{\mathbf{A}} \sum_{i\in\mathcal{N}} P_i(\mathbf{A}), \quad (16)$$

where the solution is obtained by (13). For a federation of N CSPs, the Shapley value of each CSP is obtained by calculating its average marginal contribution in all possible sub-federations $\mathcal{S} \subseteq \mathcal{N}$. Therefore, we need to know the *worth of coalition* $\upsilon(\mathcal{S}, \mathbf{A}_S)$ for all possible subsets of CSPs $\mathcal{S}$. Note that $S = |\mathcal{S}|$ and $\mathbf{A}_S$ is the corresponding $S \times S$ dimensional matrix of forwarding policies. In order to find the worth of subset $\mathcal{S}$, we have to solve the relevant optimization problem (13) for all possible such subsets.

Assuming that $\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}$, the *marginal contribution* of CSP i when it joins a sub-federation $\mathcal{S}$ is defined as

$$\mathcal{MC}_i(\mathcal{S}, \mathbf{A}_S, \upsilon) = \upsilon(\mathcal{S} \cup i, \mathbf{A}_{\mathcal{S}\cup i}) - \upsilon(\mathcal{S}, \mathbf{A}_{\mathcal{S}}) \tag{17}$$

Consequently, the profit share of a CSP i in the federation of N CSPs is given by its *Shapley value* defined as

$$\varphi_i(\mathcal{N}, \mathbf{A}) = \sum_{\mathcal{S}\subseteq\mathcal{N}\setminus\{i\}} \frac{|\mathcal{S}|!\big(N - |\mathcal{S}| - 1\big)!}{N!} \mathcal{MC}_i(\mathcal{S}, \mathbf{A}_S, \upsilon) \ , \tag{18}$$

where $\varphi_i(\mathcal{N}, \mathbf{A})$ denotes the *estimated marginal contribution* of CSP i over all possible subsets of $\mathcal{S}$.

Remark I. The two profit-sharing policies differs on how they perceived the level of a CSP's contribution. In interaction driven policy the extra profit is distributed only among the CSPs that are involved in forwarding actions of optimal policy, either as source or destination. On the other hand, Shapley value is less tight since it takes also into account the potential contribution of a CSP in all possible sub-federations. For more than two CSPs the policies may lead to totally different result.

C. Weak Federation

Weak and strong federation both require a level of commitment for each CSP in serving the requests forwarded to it by others. However, in weak federation the CSPs do not share the same objective any more, i.e. the maximization of total profit. Each of them determines its individual forwarding policy aiming to maximize its net profit, and thus a *non-cooperative game arises*. Since the CSPs have conflicting objectives, it is not sufficient to define the individual profit of each CSP as its payoff function as if the CSP were standalone. Otherwise, a selfish CSP would be able to outsource tasks without cost, taking the game to an equilibrium point where one or more CSPs may have less profit compared to that in their standalone operation. As a result, CSPs that undergo losses may be unmotivated to participate. In order to tackle this *participation constraint* and to simultaneously achieve a fair allocation of profits, it is announced to CSPs that their profit in federation is determined by a fair contribution-based profit sharing rule, namely their Shapley value. Then the CSPs are left alone to choose their own forwarding policies.

Non-cooperative Game. The set of players in this game is $\mathcal{N} = (1, 2, ..N)$. The individual forwarding strategy of a CSP i is defined by the entries of i-th row of forwarding matrix $\mathbf{A}$, thus the set of strategies of CSPs is $\mathcal{A} = (\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_N)$. Note that $\mathcal{A}$ contains the same elements as $\mathbf{A}$. We define by $\mathbf{a}_i$ the strategy of CSP i, and by $\mathbf{a}_{-i}$ the strategies of all other CSPs except i. The payoff of its CSP in the game is determined by its Shapley value, thus the set of payoffs under a set of given strategies $\mathcal{A}$ is $\varphi = (\varphi_1(\mathcal{N}, \mathcal{A}), \varphi_2(\mathcal{N}, \mathcal{A}), ..., \varphi_N(\mathcal{N}, \mathcal{A}))$.

The game starts with each CSP operating in the standalone mode, where $\mathbf{A} = \mathbf{I}$. In every step of the game, a CSP i takes as input the current forwarding policies of other CSPs $\mathbf{a}_{-i}$ and determines its best response. The best response of CSP i is to select a forwarding policy $\mathbf{a}_i$ that maximizes its payoff $\varphi_i(\mathcal{N}, \mathbf{a}_i, \mathbf{a}_{-i})$. Therefore, i determines its *best response* by solving the following optimization problem:

$$\begin{aligned} \arg\max_{\mathbf{a}_i} \quad & \varphi_i(\mathcal{N}, \mathbf{a}_i, \mathbf{a}_{-i}) \\ \text{s.t.} \quad & \alpha_{ij} \geq 0 \quad , \quad \forall j \in \mathcal{N}, \\ & \sum_{j\in\mathcal{N}} \alpha_{ij} = 1, \\ & \lambda_i^{'}(\mathbf{a}_\mathbf{i}^{\prime}) < \mu_i. \end{aligned} \tag{19}$$

In order to calculate its Shapley value, a CSP has to compute its marginal contribution in all possible sub-federations $\mathcal{S} \subseteq \mathcal{N}$. For the moment, we assume that this information is available and the game is played only for the full set $\mathcal{N}$ and not for subsets $\mathcal{S}$. At the end of this paragraph we elaborate on how the marginal contribution of CSP i in each $\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}$ can be obtained. The game continues until the system reaches a *Nash equilibrium* (NE) $\mathcal{A}^*$, where $\forall i \in \mathcal{N}$ and for every possible strategy $\mathbf{a}_i$, $\varphi_i(\mathcal{N}, \mathbf{a}_i^*, \mathbf{a}_{-i}^*) \geq \varphi_i(\mathcal{N}, \mathbf{a}_i, \mathbf{a}_{-i}^*)$.

Claim. *Given a set of forwarding strategies $\mathcal{A}$. If CSP i applies a forwarding strategy $\mathbf{a}_i^*$ that maximizes its payoff under Shapley value objective function, $\mathbf{a}_i^*$ is globally optimal.*

Proof: Given that under strategy $\mathbf{a}_i^*$ the $\varphi_i(\mathcal{N}, \mathbf{a}_i^*, \mathbf{a}_{-i}^*)$ of CPS i is maximized. Due to strong monotonicity of Shapley value [19] $\mathcal{MC}_i(\mathcal{N}, \mathbf{a}_i^*, \mathbf{a}_{-i}^*, \upsilon)$ is also maximized. From (17), $\mathcal{MC}_i(\mathcal{N}, \mathbf{a}_i^*, \mathbf{a}_{-i}^*, \upsilon)$ is maximized when the total profit of subset that i joins is maximized. Consequently, all CSPs adapt their forwarding policies in a such way that the total profit of federation is maximized.■

Corollary. *Under Shapley value payoffs the set of individually optimal forwarding strategies $\mathcal{A}^*$ is a Nash Equilibrium.*

Proof: We assume that in a step of the game all CSPs have chosen their optimal forwarding strategies $\mathcal{A}^*$ that according to our Claim are also globally optimal. We change the strategy of CSP i from $\mathbf{a}_i^*$ to $\mathbf{a}_i$, and we let the game continue. In the next step, CSP i changes back its strategy to $\mathbf{a}_i^*$ in order to maximize its payoff. Suppose that $\mathcal{A}^*$ is not a NE, there exist a CSP i that by changing its strategy to $\mathbf{a}_i$ can achieve $\varphi_i(\mathcal{N}, \mathbf{a}_i, \mathbf{a}_{-i}^*) > \varphi_i(\mathcal{N}, \mathbf{a}_i^*, \mathbf{a}_{-i}^*)$. However, since $\mathbf{a}_i^* \in \mathcal{A}^*$ this is a contradiction.■

Remark II. In order to determine its best response in previously presented game, each CSP should calculate its Shapley value based on its marginal contribution in all sub-federations $S \subseteq \mathcal{N}$. There are two alternatives to obtain this information: (i) The CSP plays recursive non-cooperative games as the above one for all the possible sub-federations. It starts playing these games from the smallest to largest subset, and the output of each game is used as input to the larger ones. (ii) Same as in section IV-B2, the CSP solves the relevant global optimization problem (13) for all subsets $\mathcal{S}$ and uses the results as input on determination of its best response in (19). Note that the second approach is less complex because we only have one game, however it has the *drawback* that CSPs should reveal their *private information* as done in strong federation.

D. Elastic Federation

The weak federation does not give CSPs the freedom to select their objective function. In this section, we propose the

elastic federation where CSPs are free to tune their level of selfishness in the federation, and thus make a choice on their objective function. The elastic federation does not require any cooperation of CSPs on exchanging private information or on deciding a fair profit sharing policy. Each CSPs advertises a price that will charge all other CSPs for serving each forwarded task. Given these, each CSP decides its individual forwarding policy aiming to maximize its profit. Again, the CSPs have conflicting objectives and thus a non-cooperative game arises. However, by setting an arbitrary price per task, there is no guarantee that in the Nash equilibrium point the individual profit of each CSP will be better than its profit in standalone operation. Thus, we provide a rule for computing prices that does attain this goal.

Inter-CSP pricing rule. This rule guarantees that each CSP i sets a price that does not violate the federation participation constraint, i.e. CSP's i profit does not decrease due to federation. This is achieved by setting a lower bound on the price of each asking. This bound is determined based on an estimate of the negative impact that a forwarded task can have on the destination CSP's profit. In addition, the pricing rule gives CSPs the freedom to be as aggressive as they wish on the selection of price. In particular, CSP i sets the price per task by following the two steps below:

1) Lower bound of price: Given that the tasks arrival rate of CSP i in standalone operation is λ_i, we estimate the profit loss that a CSP would have by accepting to serve free of charge a number of $\mu_i - \lambda_i$ more tasks so as to reach utilization factor equal to 1. The profit is affected both by the increased energy consumption cost and by QoS degradation that brings revenue losses due to price reduction. When the utilization factor reaches 1, the average completion time $d_i \to \infty$, thus the price per task (4) becomes zero and the revenue is zero. Therefore, the revenue loss that a CSP can have equals to its revenue in standalone operation $R_i(\lambda_i)$. On the other hand, the additional energy cost for serving the number of additional tasks $\mu_i - \lambda_i$ is given by subtracting its energy cost in standalone operation from the energy cost that it would have in utilization level 1. Thus, the energy loss is given by $E_i(\mu_i) - E_i(\lambda_i)$. Consequently, the profit loss of CSP i for accepting $\mu_i - \lambda_i$ more tasks without charging is given by $R_i(\lambda_i) + E_i(\mu_i) - E_i(\lambda_i)$. Consequently, we can estimate the empirical per-task average negative impact by dividing the profit loss among the number of possible additional tasks, $\mu_i - \lambda_i$. Then, CSP i can set a lower bound on price $x_i(\lambda_i)$ per task that covers its profit loss, where

$$x_i(\lambda_i) = \frac{1}{\mu_i - \lambda_i}\Big(R_i(\lambda_i) + E_i(\mu_i) - E_i(\lambda_i)\Big). \qquad (20)$$

2) Selfishness aware pricing: Having set the lower bound in the price, we now introduce the selfishness of each CSP in price setting, i.e. the level of its intrinsic desire to generate more revenue. The selfishness level of each CSP i is determined by a selfishness factor $\theta_i \in [0, 1]$, where $\theta_i = 0$ means that CSP i is not selfish and acts as being federation-friendly, and $\theta_i = 1$ implies that CSP i is extremely selfish. An extremely selfish CSP i would charge each task with a price that corresponds to the price paid by its customers in standalone operation, i.e. the price given from (4) for the current λ_i. On the contrary, a federation-friendly CSP would only charge a price $x_i(\lambda_i)$ per task.

In practice the parameter θ_i determines how the extra generated profit from a forwarded task is shared among the source and destination CSPs. If the destination CSP i is totally selfish, it gets all the extra generated profit; on the other hand if i is totally friendly, all the generated profit is gathered from the source CSP. However, being extremely selfish may discourage others from forwarding tasks toward i and select other more friendly destinations. Therefore, higher selfishness does not necessarily mean higher revenue. Based on the above analysis, the final price per task is determined as

$$\omega_i(\lambda_i) = x_i(\lambda_i) + \theta_i p_i(\lambda_i), \qquad (21)$$

where $p_i(\lambda_i)$ is the QoS-dependent pricing function (4). In this paper, we assume that θ is fixed and same for all CSPs. The selection of optimal θ_i per CSP gives rise to new game-theoretic aspects that we plan to study in the near future.

Non-cooperative Game. In elastic federation the *payoff* of a CSP i includes its individual profit, the monetary amount that i receives by charging others for serving their tasks, and the monetary amount that i pays to others for serving tasks of its customers. Hence, the payoff of CSP i is defined as

$$\psi_i(\mathbf{A}) = P_i(\mathbf{A}) + \sum_{j \in \mathcal{N} \setminus i} \alpha_{ji}\lambda_j\omega_i(\lambda_i) - \alpha_{ij}\lambda_i\omega_j(\lambda_j). \qquad (22)$$

Same as in weak federation, the set of *players* is $\mathcal{N}$ and their *strategies* are $\mathcal{A} = (\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_N)$, but now their payoff set is $\psi = (\psi_1(\mathcal{A}), \psi_2(\mathcal{A}), ..., \psi_N(\mathcal{A}))$. The *best response* of CSP i is given by the solution of $\arg\max_{\mathbf{a}_i} \psi_i(\mathbf{a}_i, \mathbf{a}_{-i})$ and under the same constraints as (19). The game stops when the CSPs converge to a Nash equilibrium point. The pure NE existence is confirmed by Debreu-Glicksberg-Fan's theorem [20].

V. Numerical Evaluation

A. Simulation setup

We focus our attention to the scenario of two CSPs in order to better understand and interpret the obtained results. We assume that the tasks that arrive in both CSP queues require an average of $L = 200$ Giga operations in order to be executed. We assume that both CSPs are symmetric with respect to computational capacity of their infrastructures $C = 2$ Tera operations per second. This capacity corresponds about 100 servers. The additional communication delay D for the forwarded tasks is taken to be an order of magnitude lower than tasks completion time in each CSP queue, $D = 0.01$. For the power consumption, we take the idle and total powers as $W_0 = 60$ KWatt and $W_1 = 400$ KWatt. Both CSPs pay the same price to their electricity provider, namely $Z = 2.7 \cdot 10^{-5}$ \$/KWatt·sec. Further, they both charge their customers according to the same pricing function, with same maximum price Q \$/task. In our experiments, we select the value of Q by taking as input the electricity price Z. In particular, given the price

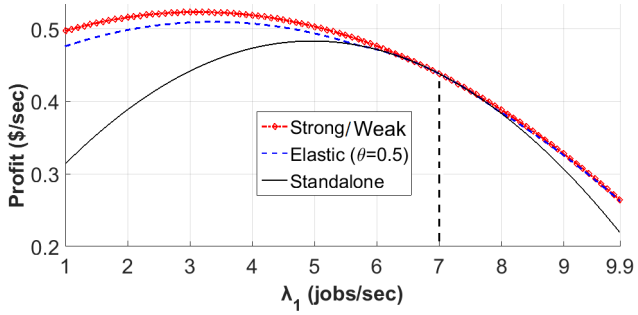


Fig. 2. Total profit of CSPs under different operation modes, for $\lambda_2 = 7$ and $\lambda_1 \in [1, 9]$

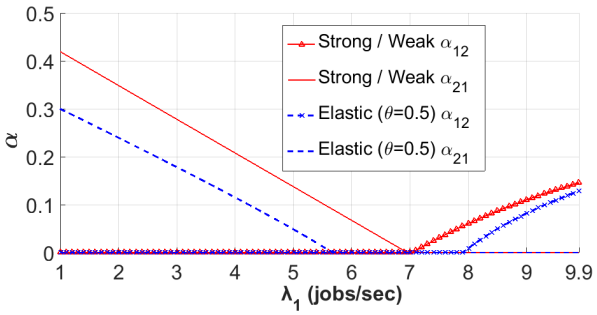


Fig. 3. Optimal forwarding policies of CSPs under different operation modes, for $\lambda_2 = 7$ and $\lambda_1 \in [1, 9]$

Z, we find the value of Q for which the profit of CSP becomes zero when the utilization factor is 0.99. This guarantees that both CSPs in standalone operation will not have negative profit for any value of utilization up to 99%. The price per task in our setup is $Q = 0.11$ \$/task. For the selfishness factor of CSPs in the case of elastic federation, we try out different combinations of θ values in the interval $[0, 1]$. In all experiments, we assume that CSP 2 has a fixed rate of incoming tasks λ_2 and we set values for λ_1 in the feasible range of values $[1, 9.9]$, with a step of 0.1. We run this type of experiment for different fixed values of λ_2 from 1 to 9.9.

B. Numerical Results

Total Profit. Fig. 2 shows the total profit under all operation modes, for fixed value of $\lambda_2 = 7$ and $\lambda_1 \in [1, 9.9]$. The results reveal that all three federation modes can achieve higher or at least the same total profit compared to the aggregate profit of CSPs in standalone operation. The total profit of strong and weak federation appears to coincide in all possible values of λ_1 and λ_2. This happens because of Shapley value's selection as a CSP's payoff in weak federation, since Shapley value urges each CSP to act for the benefit of all federation. In Fig. 2 we can observe that for $\lambda_2 = 7$ and for low load λ_1, strong and weak federation achieve a profit that is around $80-200\%$ more than the aggregate profit of CSPs in the standalone operation. For medium and high load, the benefit of strong and weak federation seems to diminish, while for $\lambda_1 = \lambda_2$, the total profit is equal to the one of standalone operation. Note that if the value of λ_2 were fixed to 9.9, the benefit of federation would be even higher for low and medium values of λ_1; about $100-400\%$ more than standalone. Consequently, the more diverse the CSPs' infrastructure utilization, the more pronounced the benefit of strong and weak federation is.

The total profit of elastic federation is strongly dependent on the selfishness factors θ of the federated CSPs. The results in Fig. 2 show that the total profit of elastic federation for $\theta = 0.5$ is lower but close enough to the one of strong and weak federation. Further, the results reveal that when θ is very close to or equal to 1, the extremely selfish CSPs set high prices and therefore the benefit of federation is eliminated. On the other hand, when θ equals to zero the total profit of elastic federation coincides with the total profit of strong and weak federation. Finally, the results show that for same value of θ, the benefit of elastic federation is relatively closer to strong and weak in high level of utilization, e.g. for $\lambda_1 = 9.9$ and $\lambda_1 = 7$ in Fig. 2.

Forwarding strategy. Fig. 3 shows the optimal forwarding policy of both CSPs under different federation modes. Interestingly, in the optimal solution at least one of α_{12} and α_{21} equals to zero. Further, the non-zero value always refers to the most utilized CSP. Strong and weak federation result to the same optimal pair $(\alpha^*_{12}, \alpha^*_{21})$. In elastic federation, the value of non-zero α parameter is affected by the selfishness factor θ of the less loaded CSP which eventually receives the forwarded tasks. If $\theta = 0$, then the optimal pair $(\alpha^*_{12}, \alpha^*_{21})$ of elastic federation is the same as in strong and weak federation. On the other hand, if the $\theta = 1$, the source CSP has no benefit from forwarding any task. Thereafter, the optimal pair of strong and weak federation is the upper bound for the optimal forwarding strategy of elastic federation. Further, we conducted additional numerical evaluations by setting different values in the communication delay D. The numerical results reveal that as the communication delay increases, the CSPs follow a more conservative forwarding policy and when D exceeds a certain value, the optimal pair becomes $(\alpha^*_{12}, \alpha^*_{21}) = (0, 0)$. Consequently, network delay is an important parameter for the effectiveness of federation.

Individual Profit. Fig. 4 and Fig. 5 show the individual profit of both CSPs under all possible operation modes. The individual profit of each CSP in all federation modes is higher or at least equal to its profit in standalone operation. The individual profit of a CSP under the interaction-driven profit sharing policy of strong federation equals its profit share when the Shapley value-driven policy is applied. However, this would be different in an experiment with more than two CSPs. In weak federation the individual profit of each CSP equals to its profit share in the strong federation. This happens because of Shapley value selection as payoff function of each CSP in the game. The individual profit of CSPs in elastic federation varies and is again related to their selfishness factor. In particular, for $\theta = 0$ a CSP that forwards a number of tasks gain all the extra revenue generated from that action, while the destination CSP only cover its profit loss. As θ increases the destination CSP demands a share of this extra generated revenue, therefore the profit share of destination CSP increases, and that of source CSP decreases. There are

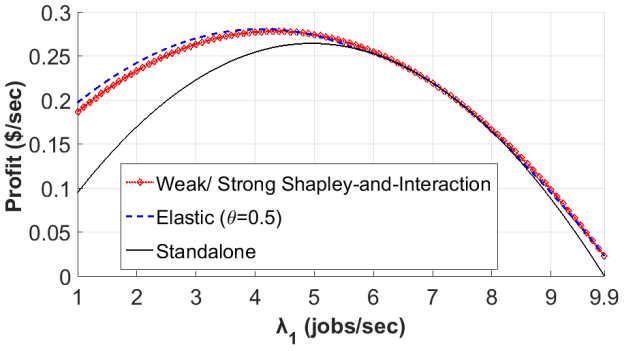


Fig. 4. Individual profit of CSP 1 under different operation modes, for $\lambda_2 = 7$ and $\lambda_1 \in [1, 9]$

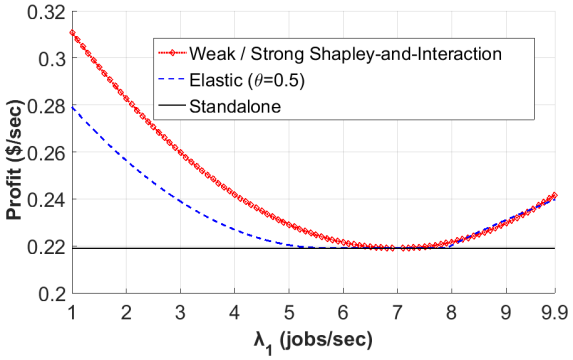


Fig. 5. Individual profit of CSP 2 under different operation modes, for $\lambda_2 = 7$ and $\lambda_1 \in [1, 9]$

non-zero values of θ where either CSP 1 or CSP 2 earns higher individual profit than in strong federation, however this cannot hold for both CSPs simultaneously because their aggregate profit cannot exceed the total profit of strong federation. A value of θ that achieves individual profit for both CSPs that are close to their profit in strong federation varies and depends on the input loads of CSPs. Consequently, the value of θ is debatable and needs further investigation.

QoS level. The results show that all federation modes outperform standalone operation and achieve a close-to-optimal QoS. The strong and weak federation achieves the same average task completion time, and further their performance is extremely close to the average completion time of a QoS-optimal federation. In elastic federation, for $\theta = 0$ the average task completion time equals the one of strong and weak federation, while for $\theta = 1$ elastic federation has the same performance as standalone operation.

VI. Conclusions

In this paper, we have presented models and policies for the formation of service-oriented cloud federations. Our models guarantee the economic sustainability of cloud federations both for cooperative and non-cooperative environments. The results show that in strong and weak federations the net profit of federation is maximized, while the offered QoS is very close the optimal one. The elastic federation gives CSPs the freedom to select their selfishness level. By selecting the appropriate selfishness level, a CSP may earn a higher individual profit than in a strong and weak federation, but the total profit of federation decreases. However, an extremely high selfishness level may deter the generation of additional individual profit.

In the present work, the forwarding policy of each CSP is consider as its strategic leverage. We plan to extend our work by investigating federation modes where the strategies of CSPs will be expressed through both forwarding policy and pricing. We also plan to study different types of federation based on an alternative model, where the federation is instantiated through computational capacity sharing instead of sharing tasks.

Acknowledgment

This work was partly funded by the Research Centre of Athens University of Economics and Business, in the framework of the project entitled "Original Scientific Publications", and by the EU Project SmartenIT (FP7-2012-ICT-317846).

References

[1] http://onapp.com/federation.
[2] http://www.arjuna.com/federation.
[3] https://www.egi.eu/infrastructure/cloud.
[4] http://www.bonfire-project.eu.
[5] http://openlab.web.cern.ch.
[6] L. Mashayekhy, M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Trans. on Cloud Computing*, vol. 3, no. 1, pp. 14–27, Jan 2015.
[7] M. Guazzone, C. Anglano, R. Aringhieri, and M. Sereno, "Distributed coalition formation in energy-aware cloud federations: A game-theoretic approach (extended version)," *CoRR*, vol. abs/1309.2444, 2013.
[8] I. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Proc. of IEEE 3rd International Conference on Cloud Computing (CLOUD)*, 2010.
[9] H. Li, C. Wu, Z. Li, and F. Lau, "Profit-maximizing virtual machine trading in a federation of selfish clouds," in *Proc. of IEEE INFOCOM*, 2013.
[10] N. Samaan, "A novel economic sharing model in a federation of selfish cloud providers," *IEEE Trans. on Parallel and Distributed Systems,*, vol. 25, no. 1, pp. 12–21, Jan 2014.
[11] G. Darzanos, I. Koutsopoulos, and G. D. Stamoulis, "A model for evaluating the economics of cloud federation," in *Proc. of 4th IEEE International Conference on Cloud Networking*, 2015.
[12] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proc. of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, 2010.
[13] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Muoz, and G. Tofetti, "Reservoir - when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, March 2011.
[14] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," in *Proc. of Cloud Computing*, 2011.
[15] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. M. Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, 2012.
[16] M. Hassan, B. Song, and E.-N. Huh, "Distributed resource allocation games in horizontal dynamic cloud federation platform," in *Proc. of IEEE 13th International Conference on High Performance Computing and Communications (HPCC)*, 2011.
[17] Y. Wang, X. Lin, and M. Pedram, "A game theoretic framework of sla-based resource allocation for competitive cloud service providers," in *Proc. of Sixth Annual IEEE Green Technologies Conference*, 2014.
[18] M. Steinder, I. Whalley, J. Hanson, and J. Kephart, "Coordinated management of power usage and runtime performance," in *Proc. Network Operations and Management Symposium (NOMS 08)*, 2008.
[19] L. S. Shapley, "A value for n-person games," Tech. Rep., 1952.
[20] G. Debreu, "A social equilibrium existence theorem," *Proc. of the National Academy of Sciences of the United States of America*, 1952.

Backward-Shifted Strategies Based on SVC for HTTP Adaptive Video Streaming

Zakaria Ye*, Rachid El-Azouzi*, Tania Jimenez*, Eitan Altman† and Stefan Valentin‡
*University of Avignon, Avignon, France, †INRIA, Paris, France, ‡Huawei Technologies, France
Email: {zakaria.ye, rachid.elazouzi, tania.jimenez}@univ-avignon.fr
eitan.altman@inria.fr, stefan.valentin@huawei.com

***Abstract*—Although HTTP-based video streaming can easily penetrate firewalls and profit from Web caches, the underlying TCP may introduce large delays in case of a sudden capacity loss. To avoid an interruption of the video stream in such cases we propose the Backward-Shifted Coding (BSC). Based on Scalable Video Coding (SVC), BSC adds a time-shifted layer of redundancy to the video stream such that future frames are downloaded at any instant. This pre-fetched content maintains a fluent video stream even under highly variant network conditions and leads to high Quality of Experience (QoE). We characterize this QoE gain by analyzing initial buffering time, re-buffering time and content resolution using the Ballot theorem. The probability generating functions of the playback interruption and of the initial buffering latency are provided in closed form. We further compute the quasi-stationary distribution of the video quality, in order to compute the average quality, as well as temporal variability in video quality. Employing these analytic results to optimize QoE shows interesting trade-offs and video streaming at outstanding fluency.**

***Index Terms*—Scalable Video Coding, Quality of Experience, Queuing Theory, Ballot theorem, QoE Optimization**

I. Introduction

Recent studies show that video streaming already generated 45% of all mobile data traffic in 2014 [1]. By 2019, this fraction will likely increase to 62%, while an 11-fold increase is predicted for the overall mobile data traffic [1]. Despite the recent advances in increasing wireless capacity, this massive traffic load will drive mobile networks further into saturation and will turn user satisfaction into an enormous challenge. Consequently, more and more content providers and network operators will focus on Quality of Experience (QoE) per unit cost as primary metric for operational efficiency [2].

The dominating factors of QoE are widely studied [3, 4]. Recent works developed approaches for understanding user engagements metrics [5, 7]. The direct relation between time spent in rebuffering and user engagement was shown in [5]. In [7], the buffering ratio, rate of buffering, start up delay, rendering quality and average bit rate were demonstrated to show a dominating effect on QoE. Guidance to operators for improving user engagement in real time using only network-side measurements is provided in [8]. Authors of [9] found that temporal quality variation is worse than keeping a constant quality that is lower on the average.

In general, understanding the QoE of mobile video is a complex task due to the many relationships between metrics and end-user's perceived video quality, metric-to-metric dependencies and confounding factors [8]. At the same time, highly dynamic load and channel states lead to fluctuating capacity not only in mobile networks.

To efficiently trade-off fluency and visual quality, more and more content providers deploy HTTP Adaptive Streaming (HAS) solutions, which are standardized as MPEG Dynamic Adaptive Streaming over HTTP (DASH) [10]. With DASH, each video file is divided into multiple small segments and each segment is encoded into multiple quality levels. Based on the available capacity, the client adaptively chooses the quality level of the segment such that visual quality is maximized at a low risk for an empty playback buffer.

The video segments can be created by encoding video content with various compression algorithms, where those following H.264/AVC (Advanced Video Coding) and H.264/SVC (Scalable video coding) [12, 13] are widely employed. Although each AVC encoding run generates only segments of a single bitrate, segments for various bitrates can be created in multiple runs and chosen adaptively with HAS. This leads to a multiple bitrate video stream even with a non-scalable technique as AVC. On the other hand, SVC directly supports multiple bitrates within a single segment by multi-layer coding. With this technique, the video stream is encoded in one base layer and one or more enhancement layers. The base layer is always requested and, at sufficient capacity, one or more enhancement layers are additionally requested.

A. Related Literature

One benefit of HAS video streaming is that HTTP traffic can easily penetrate firewalls and profit from Web-infrastructure such as proxies and Content Delivery Networks (CDNs). The drawback, however, is that the underlying TCP protocol may introduce substantial delays to cope with packet errors and contention. For the video, this means that pixels errors and frame drops can essentially be ignored while resolution, initial buffering latency, starvation duration, and rate of buffering become the dominating factors for QoE. Such latencies are the result of an empty playback buffer as a consequence of choosing a higher quality than the supported bitrate [15]. To avoid such erroneous adaptation, current HAS policies are based on the measured segment fetch time that allows an instantaneous adaptation [17]. Authors in [16] developed a rate adaptive method to enhance DASH performance over

ISBN 978-3-901882-83-8

multiple content distribution servers. A Fuzzy-based controller has been proposed to dynamically adapt the video bitrate based on both the estimated throughput and the size of the playback buffer [20]. Other approaches have been explored by jointly considering the characteristics of the media content and the available wireless resources in the operator network [18, 19].

This paper proposes a complementary solution to DASH, named Backward-Shifted Coding (BSC). This solution makes HAS more robust to rapid fluctuations of the network capacity and provides more flexibility in increasing the quality of video without playback interruption. Furthermore, we develop an exact approach to obtain the distributions of the number of the playback interruptions and of the initial buffering latency. This analysis is closed to that of M/M/1 queue model in [22] and allows us to obtain an explicit formulation for the QoE metrics. While the bounds on the playback interruption probability were obtained in [21] for an M/D/1 queue, our paper provides a new analysis for BSC which also obtains the average quality and temporal variability in video quality by using the quasi-stationary regime.

B. Main Contributions

This paper provides important insight in optimization of HTTP adaptive video streaming and proposes a new Backward-Shifted Coding (BSC) scheme. The main idea of BSC is to add a layer of time-shifted redundancy to the video stream such that future frames are downloaded at any instant. In case of a sudden capacity drop, these pre-fetched frames can be played back and maintain a smooth video stream at sufficient quality. BSC can be implemented using the standard codecs and allows us to analytically obtain the key QoE factors by using the Ballot theorem. Using these factors as inputs we finally propose a QoE optimization function.

We can summarize the main contributions of this paper as follows:

1. We develop a novel coding scheme to improve the user QoE in HTTP adaptive streaming.
2. We present explicit form expressions for the QoE metrics.
3. We propose an optimization scheme that take into account not only the waiting time, but also the mean video quality.
4. We show that our scheme can render a better QoE than existing bitrate adaptation algorithms used in DASH.
5. We show that our scheme can greatly reduce the probability of video playback interruption using several frames arrivals processes.

C. Paper Organization

The remainder of this paper is structured as follows: Section II describes the system model. Section III presents the analytical model for computing the QoE metrics using the Ballot theorem. Section IV presents the optimization issue for the QoE metrics while Section V verifies the theoretical results and shows some numerical examples. Section VI concludes the paper.

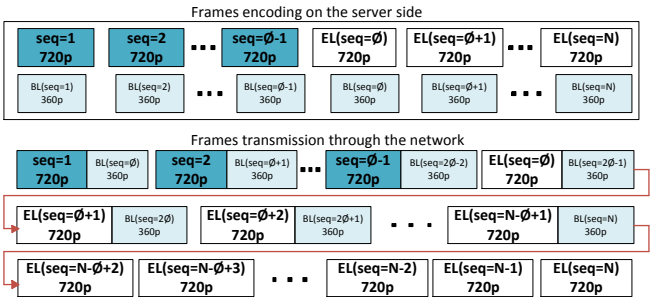


Fig. 1: Using SVC in Backward-Shifted Coding

II. System Design

In this section we describe how our scheme Backward-Shifted Coding (BSC) can be used with any video codec that follows the H.264/SVC standard [13]. Then we describe the integration into HAS based on the MPEG-DASH standard [10].

A. Mapping from BSC Scheme to Coding Scheme

BSC is entirely client driven and it is independent of the video compression standard. The main idea of this scheme is to shift the base layer frames (low quality) and the enhancement layer frames (optimal quality), so that, when an interruption of playback buffer occurs, the base layer frames can still be played. To each frame n, we add its base layer in some subsequent frame $n-\phi+1$. If the starvation happened at frame n, the playback retrieves the base layer frame from frame $n-\phi+1$. In particular, we exploit temporal redundancy between subsequent frames in order to avoid the interruption of the playback buffer. Our scheme is inspired from Forward Error Correction (FEC) where the encoder adds redundancy to a message. However, using the SVC codec, the BSC scheme does not generate any overhead or redundant frames compared to the FEC scheme. Indeed, with an SVC codec, the video bitstream contains a base layer and number of enhancement layers. The enhancement layers are added to the base layer to further improve the quality of video by increasing video frame-rate, temporal and quality scalability and spatial resolution. In our scheme, base layer and enhancements layers are temporally shifted as shown in Figure 1. We assume ϕ to be the offset between the basic layer frame and its enhancement layers. Frame n with $1 \leq n \leq \phi-1$ contains two blocks: complete block n and the base layer of block $n+\phi-1$. Frame n with $n > \phi-1$ contains two blocks: the enhancement layers of block n and the base layer of block $n+\phi-1$. At the user side, incoming bits are reassembled into video frames by the decoder. Starvation under BSC can happen at block n if the base layer block n is missing and the quality switching occurs when the enhancement layer is missing and the player finds only the base layer block n.

B. Mapping from BSC Scheme to DASH

In DASH systems, each video consists of multiple small segments at the media server and each segment is encoded

at multiple discrete bitrates. Hence, the BSC scheme can be used at client side with DASH. In fact, when using the BSC scheme with DASH, the video player decides the most suitable quality level of the base layer and the number of enhancement layers. For example, given three video resolutions 720p, 480p and 360p, the video player can decide the most suitable configuration between the base layer and enhancement layer such as 360p (BL) and 480p (1 EL) or 360p (BL) and 720p (2 ELs). This gives more flexibility to the video player to detect the highest quality level the current network conditions can support.

III. Mathematical Performance Evaluation

In order to evaluate the efficiency of BSC using the SVC codec, in this section, we develop a novel performance evaluation for QoE based on the Ballot theorem [11]. This analysis captures QoE factors such as fraction of time spent rebuffering, content resolution and initial buffering latency. The analysis of starvation is closely related to analyzing the busy period in transient queues but differs in two aspects: First, our work aims to find the probability generating function of starvation events and not the queue size. Second, we do not assume a stationary arrival process.

A. Starvation Analysis

In this section, we call optimal frame the enhancement layers. The non-optimal frame corresponds to the base layer. We assume N to be the media file size. When the streaming packets traverse the network, their arrivals to the media player are not deterministic due to the dynamic of the available bandwidth. The packets are reassembled by the decoder to render the video frames. We assume a Poisson distribution to describe the frames arrivals. After the streaming frames are received, they are first stored in the playout buffer. The interval between the service of two frames is assumed to be exponentially distributed so that we can model the receiver buffer as an M/M/1 queue. The exponential distributed assumption is not the most realistic way to describe frame arrivals, but it reveals the essential features of the system, and it is the first step for more general arrival processes. In Section V we evaluate the performance of the Backward-Shifted Coding system by simulation, using different types of packet arrivals process such as the logistic process and the on-off process. The logistic process fits the video streaming traffic on the Long Term Evolution (LTE) networks according to [14]

The maximum buffer size is assumed to be large enough to exclude buffer overflows. ϕ is the offset between the optimal frame and its corresponding non-optimal frame (Fig. 2). A starvation happens when the playout buffer is empty.

We denote by λ the Poisson arrival rate of the frames, and by μ the Poisson service rate. We define $\rho = \lambda/\mu$ to be the traffic load.

In a non-empty M/M/1 queue with everlasting arrivals, the rate at which either an arrival or a departure occurs is given by $\lambda+\mu$. This event corresponds to an arrival with probability p, or is otherwise to an end of service with probability q, where

$$p = \frac{\lambda}{\lambda+\mu} = \frac{\rho}{1+\rho}; \qquad q = \frac{\mu}{\lambda+\mu} = \frac{1}{1+\rho}$$

The buffer is initially empty. We let T_x be the initial buffering delay, in which x frames are accumulated in the buffer.

*1) **Probability of Starvation**:* We present a frame level model to investigate the starvation probability with the BSC. Our analysis of the probability of starvation is built on the Ballot theorem[1].

Since we set the value of ϕ at the beginning of the video session, the starvation can happen before the arrival of frame ϕ if $\phi > x$. So we have to investigate the probability of starvation by distinguishing the two cases: $\phi \leq x$ and $\phi > x$. Let $P_s^<(N,\phi,x)$ and $P_s^>(N,\phi,x)$ denote, respectively, the probability of starvation for $\phi \leq x$ and $\phi > x$. For $\phi \leq x$, the media player starts to work when the number of optimal frames in the buffer reaches x optimal frames which corresponds to $x+\phi-1$ non-optimal frames stored in the buffer. Thus $P_s^<(N,\phi,x)$ is given by [22],

$$P_s^<(N,\phi,x) = \sum_{k=x+\phi-1}^{N-1} \frac{x+\phi-1}{2k-x-\phi+1}\binom{2k-x-\phi+1}{k-x-\phi+1}\cdot p^{k-x-\phi+1}q^k. \quad (1)$$

Let $P_{x,\phi}^k$ and P_x^k denote, respectively, the probability that the starvation happens exactly after the departure of frame k using BSC and without BSC. These probabilities are given by

$$P_{x,\phi}^k = \frac{x+\phi-1}{2k-x-\phi+1}\binom{2k-x-\phi+1}{k-x-\phi+1}\cdot p^{k-x-\phi+1}q^k,$$
$$P_x^k = \frac{x}{2k-x}\binom{2k-x}{k-x}\cdot p^{k-x}\cdot q^k. \quad (2)$$

Theorem 1. *For the offset $\phi > x$, the probability of starvation is given by:*

$$P_s^>(N,\phi,x) = P_{s1} + (1 - P_{s1}) \cdot P_{s2} \quad (3)$$

where

$$P_{s1} = \sum_{k=x}^{\phi-2} \frac{x}{2k-x}\binom{2k-x}{k-x}\cdot p^{k-x}\cdot q^k, \quad (4)$$

and

$$P_{s2} = \sum_{k=2\phi-2}^{N-1} \frac{x+\phi-1}{2k-x-\phi+1}\binom{2k-x-\phi+1}{k-x-\phi+1}\cdot p^{k-x-\phi+1}q^k. \quad (5)$$

Proof. For the case $\phi > x$, the starvation could happen before the arrival of frame ϕ. We define $E_{<\phi}$ and $E_{>\phi}$ to be the event that the starvation happens for the first time before ϕ and after

[1]Ballot theorem: In a ballot, candidate A scores N_A votes and candidate B scores N_B votes, where $N_A > N_B$. Assume that while counting, all the ordering (i.e. all sequences of A's and B's) are equally alike, the probability that throughout the counting, A is always ahead in the count of votes is $\frac{N_A-N_B}{N_A+N_B}$.

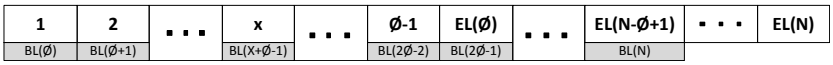


Fig. 2: The BSC Coding for the offset $\phi > x$. The starvation can happen before the arrival of optimal frame ϕ since $x < \phi$.

ϕ, respectively. The event of starvation is $E_{<\phi} \cup (E_{>\phi} \cap \bar{E}_{<\phi})$; where $\bar{E}_{<\phi}$ is the complementary of $E_{<\phi}$ and is the event that no starvation happens before the arrival of frame ϕ. We have $P(E_{<\phi} \cup (E_{>\phi} \cap \bar{E}_{<\phi})) = P(E_{<\phi}) + P(E_{>\phi} \cap \bar{E}_{<\phi})$. For the event $E_{<\phi}$, since we cannot use the non-optimal frames of the BSC coding because the starvation happens before ϕ, the probability of starvation is P_{s1}. We exclude in this sum $\phi - 1$ since the starvation cannot happen after the service of frame $\phi - 1$ because the frame ϕ (non-optimal) is already stored in the buffer. Now we compute the probability of the second term $P(E_{>\phi} \cap \bar{E}_{<\phi})$. We have $P(E_{>\phi} \cap \bar{E}_{<\phi}) = P(E_{>\phi}/\bar{E}_{<\phi})P(\bar{E}_{<\phi})$. Since $\bar{E}_{<\phi}$ is the complementary of $E_{<\phi}$, $P(\bar{E}_{<\phi})$ is $1 - P_{s1}$. $E_{>\phi}/\bar{E}_{<\phi}$ is the event that a starvation happens for the first time after the arrival of frame $\phi - 1$, given that the starvation does not happen before. Then, assuming that the starvation does not happen before $\phi - 1$, frame $x + \phi - 1$ will be played after $\phi - 1$ as shown in Fig. 3. At this instant, the non-optimal frame $2\phi - 2$ is already in the buffer, so the starvation cannot happen before the service of $2\phi - 2$. We use the Ballot theorem to compute the probability of the event $E_{>\phi}/\bar{E}_{<\phi}$ based on the non-optimal frames. The most important trick is the origin of the Ballot theorem, i.e., where to start the process of counting the frames arrivals and departures. The inappropriate method is to start the counting process just after the arrival of the frame $\phi - 1$. At that moment, we do not know the number of departures that occur before. So we start the counting process when we have x optimal frames in the buffer, that correspond to the last non-optimal frame $x + \phi - 1$. We define A_k to be an event that the buffer becomes empty for the first time when the service of frame k is finished (Fig. 3). All the events $A_k, k = 1, ...N$, are mutually exclusive. The event of starvation is the union $\cup_{k=2\phi-2}^{N-1} A_k$. We exclude in this union A_k for $k \in [1, 2\phi - 3]$ because we cannot have a starvation before the arrival of optimal frame $\phi - 1$. That corresponds to the non-optimal frame $2\phi - 2$. This union of events excludes E_N because the empty buffer after the service of N packets is not a starvation. When the buffer is empty at the end of the service of the k^{th} packet, the number of arrivals is $k - x - \phi + 1$ after the pre-fetching process. The probability of having $k - x - \phi + 1$ arrivals and k departures is computed from the binomial distribution $\binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} \cdot q^k$. For the necessary and sufficient condition of the event A_k, we apply the Ballot theorem. If we count the number of arrivals and departures when the playback starts, the number of departures is always greater than the number of arrivals. Otherwise, the empty buffer already happens before the k^{th} frame is served. According to the Ballot theorem, the probability of event A_k is computed by $\frac{x+\phi-1}{2k-x-\phi+1}\binom{2k-x-\phi+1}{k-x-\phi+1} \cdot p^{k-x-\phi+1} \cdot q^k$. Therefore, the probability of the event $E_{>\phi}/\bar{E}_{<\phi}$, is the probability of the union $\cup_{k=x+\phi-1}^{N-1} A_k$, given by (5). □

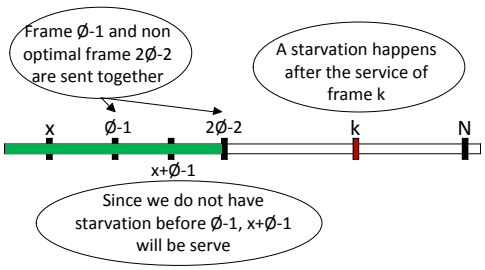


Fig. 3: If the starvation does not happen before $\phi - 1$, then there will be no starvation untill the service of frame $2\phi - 2$.

*2) **Probability Generating Function of Starvation Events**:* In the BSC scheme, the starvation may happen for more than once during the file transfer. We are interested in the probability distribution of starvation, given the finite file size N. When $\phi \leq x$, we cannot have a starvation before the service of $\phi - 1$. In this case, the probability generating function is similar to that of [22] in replacing x by $x + \phi - 1$. Now, we show how the probability generating function of starvation events can be derived using the Ballot theorem for the case $\phi > x$. We define a path as a complete sequence of frame arrivals and departures [22]. The probability of a path depends on the number of starvations. We consider a path with j starvations. To carry out the analysis, we start from the event that the first starvation takes place. We denote by k_l the l^{th} departure of a frame that sees an empty queue. We notice that the path can be decomposed into the following mutually exclusive events:

- Event $\mathcal{F}(k_1)$: the buffer becoming empty for the first time in the entire path.
- Event $\mathcal{M}_l(k_l, k_{l+1})$: the empty buffer after the service of frame k_{l+1} given that the previous empty buffer happens at the departure of frame k_l.
- Event $\mathcal{L}_j(k_j)$: the last empty buffer observed after the departure of the last frame k_j.

We let $P_{\mathcal{F}(k_1)}$, $P_{\mathcal{M}_l(k_l,k_{l+1})}$ and $P_{\mathcal{L}_j(k_j)}$ be the probabilities of events $\mathcal{F}(k_1)$, $\mathcal{M}_l(k_l, k_{l+1})$ and $\mathcal{L}_j(k_j)$, respectively, and will analyze the probabilities of these events step by step. We first compute the probability of having only one starvation. This probability concerns the two events: $\mathcal{F}(k_1)$ and $\mathcal{L}_1(k_1)$,

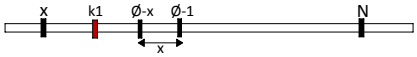


Fig. 4: A starvation happens at $k_1 < \phi - x$.

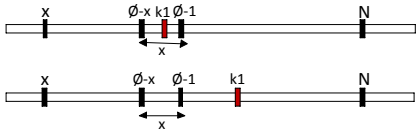


Fig. 5: A starvation happens at $k_1 > \phi - x$.

i.e., the event that the buffer becomes empty for the first time after the service of the frame k_1 and the event that we do not observe an empty buffer after k_1 until the end of the video file (Fig. 4 and 5). The starvation can happen at k_1 before we use the BSC redundant frames, i.e., before the arrival of frame $\phi - 1$. In this case, the probability of starvation is given by $P_x^{k_1}$ of (2). If the starvation happens after the arrival of frame $\phi - 1$, then it necessarily happens after the service of frame $2\phi - 3$ since we cannot have a starvation between $\phi - 2$ and $2\phi - 3$. Then the probability of starvation is given by Theorem 1. So the probability distribution of event $\mathcal{F}(k_1)$ is expressed as

$$P_{\mathcal{F}(k_1)} := \begin{cases} 0, & if \;\; k_1 < x \;\; or \;\; k_1 = N; \\ P_x^{k_1} & if \;\; k_1 \in [x, ..., \phi - 2]; \\ 0, & if \;\; k_1 \in [\phi - 1, ..., 2\phi - 3]; \\ (1 - P_{s1}) P_{x,\phi}^{k_1}, & \\ if \;\; k_1 \in [2\phi - 2, ..., N - 1]. & \end{cases} \quad (6)$$

$P_x^{k_1}$, $P_{x,\phi}^{k_1}$ and P_{s1} are given by (2) and (4). Given that the only starvation happens at k_1, what is the probability that no starvation happens until the end of the video? That is the probability of the event $\mathcal{L}_1(k_1)$. We take the complement of starvation probability as the probability of no starvation for the file size $N - k_1$. We denote $x_\phi = x + \phi - 1$ to simplify the expressions. We distinguish two cases (Fig. 4 and 5). The first case is that the starvation happens at k_1, and we still can have a starvation before the arrival of frame $\phi - 1$ (Fig. 4). Then, we use the probability of starvation of the theorem 1, $P_s^{>}(N - k_1, \phi, x)$ to compute the probability of having no starvation until the end of the video file. For the remaining case (Fig. 5), a starvation cannot happen before we use the non-optimal frames. Then, we use the probability of starvation, $P_s^{<}(N - k_1, \phi, x)$. Finally, the probability distribution of event $\mathcal{L}_1(k_1)$ is expressed by

$$P_{\mathcal{L}_1(k_1)} = \begin{cases} 0, & if \;\; k_1 < x \;\; or \;\; k_1 = N; \\ 1 - P_s^{>}(N - k_1, \phi, x), & if \;\; x \le k_1 < \phi - x; \\ 1 - P_s^{<}(N - k_1, \phi, x), & if \\ \phi - x \le k_1 < \phi - 1 \;\; or \;\; 2\phi - 2 \le k_1 < N - x_\phi; & \\ 0, & if \;\; \phi - 1 \le k_1 < 2\phi - 3; \\ 1, & if \;\; N - x_\phi \le k_1 < N. \end{cases} \quad (7)$$

We denote by $P_s(j)$ the probability of having j starvations. For the case with one starvation, $P_s(1)$ is solved by

$$P_s(1) = \sum_{i=1}^{N} P_{\mathcal{F}(i)} P_{\mathcal{L}_1(i)} = \mathbf{P}_{\mathcal{F}} \cdot \mathbf{P}_{\mathcal{L}_1}^T \quad (8)$$

where ^{T} denotes the transpose. Here, $\mathbf{P}_{\mathcal{F}}$ is the row vector of $P_{\mathcal{F}(i)}$, and $\mathbf{P}_{\mathcal{L}_1}$ is the row vector of $P_{\mathcal{L}_1(i)}$, for $i = 1, 2, ..., N$.

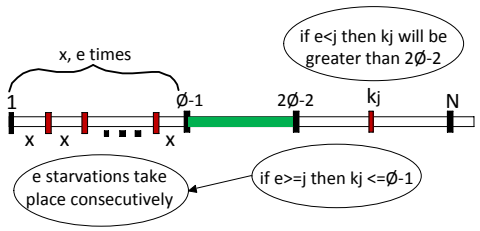


Fig. 6: The lower bound of k_j in case we have j starvations.

Now we compute the probability of having more than one starvation. A path with j starvations is composed of a succession of events

$$\mathcal{F}(k_1), \mathcal{M}_1(k_1, k_2), ..., \mathcal{M}_{j-1}(k_{j-1}, k_j), \mathcal{L}_j(k_j).$$

We have to solve the probability distribution of the events $\mathcal{L}_j(k_j)$ and $\mathcal{M}_l(k_l, k_{l+1})$. Suppose that there are j starvations after the service of frame k_j. Then, only the lower bound of k_j changes in the event $\mathcal{L}_j(k_j)$ from $\mathcal{L}_1(k_1)$. This lower bound corresponds to the extreme case where the j starvations take place consecutively. Let e be the number of starvations that we can have before frame $\phi - 1$ in the extreme case. So $e = \lfloor \frac{\phi - 2}{x} \rfloor$. We distinguish two cases where $e \ge j$ and $e < j$ (Fig. 6). If $e \ge j$, then all the j starvations will happen before the service of frame $\phi - 1$ and the lower bound of k_j is jx. Then we find the expression of $\mathcal{L}_j(k_j)$ in replacing the lower bound x by jx in the expression of $\mathcal{L}_1(k_1)$. If $e < j$, then the remaining starvations will happen after the service of frame $2\phi - 2$ since we do not have starvation between $\phi - 1$ and $2\phi - 2$. Hence, the probability distribution of event $\mathcal{L}_j(k_j)$ is

$$P_{\mathcal{L}_j(k_j)} = \begin{cases} 0, & \\ if \;\; k_j < 2\phi - 2 + (j - e)x_\phi \;\; or \;\; k_j = N; & \\ 1 - P_s^{<}(N - k_j, \phi, x), & \\ if \;\; 2\phi - 2 + (j - e)x_\phi \le k_j < N - x_\phi; & \\ 1, \quad if \quad N - x_\phi \le k_j < N. & \end{cases} \quad (9)$$

We now compute the probability of the event $\mathcal{M}_l(k_l, k_{l+1})$. After frame k_l is served, the l^{th} starvation is observed. We compare k_l to e to obtain the position of k_l. If $e < l$, then k_l should not be less than $2\phi - 2 + (l - e)(x + \phi - 1)$ in order to have l starvations. Also, k_{l+1} must satisfy $k_l + (x + \phi - 1) \le k_{l+1} < N - (j - l - 1)(x + \phi - 1)$ because of the pre-fetching after k_l and the fact that we have j starvations in total. In this case, $P_{\mathcal{M}_l(k_l, k_{l+1})}$ is expressed as $\frac{x+\phi-1}{2k_{l+1}-2k_l-x-\phi+1} \binom{2k_{l+1}-2k_l-x-\phi+1}{k_{l+1}-k_l-x-\phi+1} p^{k_{l+1}-k_l-x-\phi+1} q^{k_{l+1}-k_l}$. If $e \ge l$, then we have $lx \le k_l \le \phi - 1$. For $k_l + x \le k_{l+1} < \phi - 1$, $P_{\mathcal{M}_l(k_l, k_{l+1})}$ is expressed as $\frac{x}{2k_{l+1}-2k_l-x} \binom{2k_{l+1}-2k_l-x}{k_{l+1}-k_l-x} p^{k_{l+1}-k_l-x} q^{k_{l+1}-k_l}$. Since the $(l+1)^{th}$ starvation cannot happen between $\phi - 1$ and $2\phi - 2$, for $2\phi - 2 \le k_{l+1} < N - (j - l - 1)(x + \phi - 1)$, the probability of the event $\mathcal{M}_l(k_l, k_{l+1})$ is expressed as the probability for the case $e < l$. We denote by $\mathbf{P}_{\mathcal{M}_l}$ the matrix of $P_{\mathcal{M}_l(k_l, k_{l+1})}$ for $k_l, k_{l+1} \in [1, N]$. The probability of having $j (j \ge 2)$ starvations is given by

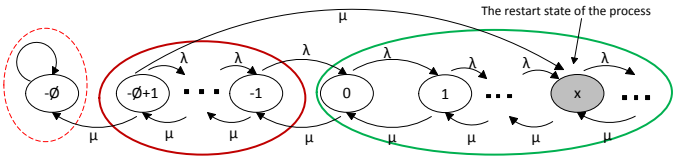


Fig. 7: Markov model of the switching mechanism, denoting the difference between the number of optimal and non-optimal frames in the playout buffer.

$$P_s(j) = \sum_{k_1=1}^{N}\sum_{k_2=1}^{N},\ldots,\sum_{k_{j-1}=1}^{N}\sum_{k_j=1}^{N} P_{\mathcal{F}(k_1)}\cdot P_{\mathcal{M}_1(k_1,k_2)},\ldots,$$

$$P_{\mathcal{M}_{j-1}(k_{j-1},k_j)}\cdot P_{\mathcal{L}_j(k_j)} = \mathbf{P}_{\mathcal{F}}\cdot\left(\prod_{l=1}^{j-1}\mathbf{P}_{\mathcal{M}_l}\right)\cdot\mathbf{P}^T_{\mathcal{L}_j}. \quad (10)$$

Then, we can write the probability generating function (p.g.f) $G(z)$ by

$$G(z) = E(z^j) = \sum_{j=0}^{J} P_s(j)\cdot z^j. \quad (11)$$

B. Analysis of the Video Quality

In this section, we analyze the average video quality of the BSC system. For this purpose we model the system as a continuous-time Markov birth-death process with an absorbing state which corresponds to the starvation event. Then, we compute the amount of time spent in each bitrate level. We call low bitrate, the bitrate of the base layer frames (or non-optimal frames) and optimal bitrate, the bitrate of the combined base and enhancement layers frames. The switching process is shown in Fig. 7.

Let A and D be the sequence number of the last optimal frame in the buffer and the sequence number of the last frame that was displayed at the screen respectively. The state n of the Markov process is $A-D$. If $D \leq A$, the state n is positive and there is exactly $A-D$ available optimal frames in the buffer. Otherwise, if $D > A$, the state n is negative. In that case, there is no more available optimal frames and the number of available non optimal frames is $A-D+\phi$. For example if the process is in the state $-\phi+1$ (i.e., $A-D=-\phi+1$), it remains exactly 1 non optimal frame in the buffer as shown in Fig. 7.

The infinitesimal generator Q of the process is a tridiagonal matrix where elements of the diagonal, upon the diagonal and under the diagonal are $-(\lambda+\mu)$, λ and μ respectively, except the first element of the diagonal and upon the diagonal which is 0 (due to the absorbing state).

First, we compute the time to absorption starting from initial state $k > -\phi$. To do this, we substitute transition to the absorbing state $-\phi$ by transition to the initial state k, whenever the resulting Markov chain is ergodic and admits a stationary regime $\mathbf{q} = (q_{-\phi+1}, q_{-\phi+2}, ..)$. $\mathbf{q}$ is given by solving the balance equations of the resulting Markov chain

$$\begin{cases} \mu q_{-\phi+2} &= (\lambda+\mu)q_{-\phi+1} \\ \lambda q_{j-1}+\mu q_{j+1} &= (\lambda+\mu)q_j, j\neq x+1 \\ \lambda q_{x-1}+\mu(q_{x+1}+q_{-\phi+1}) &= (\lambda+\mu)q_x \\ \sum_{j=-\phi+1}^{\infty} q_j = 1 \end{cases} \quad (12)$$

$$q_j = q_{-\phi+1}\frac{1-\rho^{j+\phi}}{1-\rho},\ \ j=-\phi+1,-\phi+2,..,x$$

$$q_j = q_{-\phi+1}\frac{1-\rho^{x+\phi}}{1-\rho}\rho^{j-x},\ \ j=x+1,x+2,...$$

Thus, we have

$$\begin{cases} q_{-\phi+1} &= \frac{1-\rho}{\phi+x} \\ q_j &= \frac{(1-\rho^{j+\phi})}{(\phi+x)},\ \ j=-\phi+1,..,x \\ q_j &= \frac{(1-\rho^{x+\phi})}{((\phi+x)}\rho^{j-x},\ \ j=x+1,x+2,... \end{cases} \quad (13)$$

Let $E[S_i]$ be the expected time spent in state i before the process reaches the absorbing state. From [6], we have

$$E[S_i] = q_i E[\tau_x]$$

where $E_{[\tau_x]}$ is the time to absorption into state $-\phi$ starting from the initial state x. According to [27], the mean time to absorption into state $-\phi$ from the initial state x is given by

$$E[\tau_x] = \frac{x+\phi}{\mu-\lambda} \quad (14)$$

Thus the expected time spent in state i before the process reaches the absorbing state is given by

$$E[S_i] = q_i\frac{x+\phi}{\mu-\lambda} \quad (15)$$

where q_i is given by (13). Thus the time spent in the low bitrate and in the optimal bitrate are given, respectively, by

$$T_{\mathcal{L}} = \sum_{i=-\phi+1}^{-1} E[S_i] \text{ and } T_{\mathcal{H}} = \sum_{i=0}^{\infty} E[S_i] \quad (16)$$

Let $b_{\mathcal{L}}$ and $b_{\mathcal{H}}$ be the bitrate of the low coding (base layer) and the bitrate of the optimal coding (base and enhancement layers) respectively. Hence, the average bitrate is

$$b_{avg} = \frac{T_{\mathcal{L}}b_{\mathcal{L}} + b_{\mathcal{H}}T_{\mathcal{H}}}{T_{\mathcal{L}}+T_{\mathcal{H}}} \quad (17)$$

Let us now compute the distribution of period of time during which the video playback quality is optimal, named B_H. This period corresponds to the duration of time that the process starting from state 1, stays continuously away from state 0. Using the analysis from the busy period in M/M/1, the expected and variance of B_H is given, respectively, by [27]

$$E[B_H] = \frac{1}{\mu-\lambda}, \quad \text{and } V(B_H) = \frac{(1+\rho)}{\mu^2(1-\rho)^3}$$

The analysis of the variation of the quality is omitted in this paper due to lack of space.

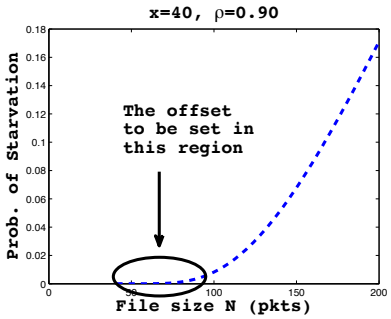


Fig. 8: The value of the offset ϕ.

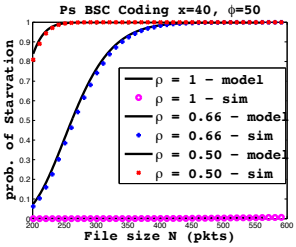


Fig. 9: The probability of starvation vs the file size N

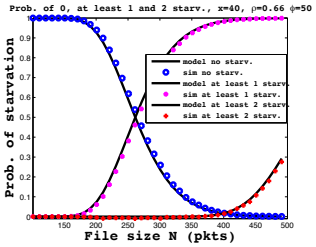


Fig. 10: The probability of no starvation, or having at least one and two starvations vs the file size N

C. The Initial Buffering Delay and Rebuffering Delay

The expected initial buffering delay is $\frac{x}{\lambda}$ and the expected rebuffering delay is $\frac{x+\phi-1}{\lambda}$. Note that the BSC scheme increases the start-up delay compared to an equivalent single bitrate system because of the shift ϕ. Indeed, we send the enhancement layer ϕ after the complete frame $\phi - 1$. But in practice, this delay does not exceed a couple of seconds, then it does not have a huge impact on the overall quality of experience. However, ϕ impacts the rebuffering delay, i.e., the waiting time after a starvation event. The player starts when we accumulate x optimal frames in the buffer. So a high value of ϕ increases the amount of rebuffering time.

IV. Evaluation of the QoE

In this section, we show how the BSC can be used to improve the metrics of the quality of experience. The parameter ϕ affects the initial buffering delay and the starvation. We first confirm this with the model and show how one can choose ϕ to improve the QoE metrics. Then, we propose an evaluation function to improve the user experience during the video session.

A. Choosing the Offset ϕ

To set the value of the offset ϕ, we should know the probability of starvation of a simple system without BSC according to the file size N. This probability is given in [22] by

$$P_{starvation} = \sum_{k=x}^{N-1} \frac{x}{2k-x} \binom{2k-x}{k-x} \cdot p^{k-x} \cdot q^k, \qquad (18)$$

It also corresponds to the probability of playing the non optimal frames for the first time in BSC system since a starvation in a system without BSC is equivalent to a switching in BSC system. The surrounded region in figure 8 corresonds to the set of values of N where the risk of the playback interruption is small (ϕ can be set up to 80 in the figure without risk of starvation). Then, we can choose the offset ϕ to improve the QoE metrics: the rebuffering time, the starvation and the average bitrate. We have less risk of starvation for some values of ϕ. Since after each starvation event, we wait for $x+\phi-1$ frames, small values of ϕ decreases the rebuffering time. Moreover, (16) shows that the time spent in low bitrate level increases with ϕ. Then, ϕ must satisfy these QoE metrics requirements too. Parameter ϕ is calculated at the beginning of the video session based on the networks conditions and the user profile but its value can be updated after each starvation if the networks conditions changed.

B. Evaluation Scheme for the Global QoE

To use our analysis for optimizing the QoE, we define an objective QoE cost function $C(x, N)$ for a user, which includes the expected initial buffering latency, the number of playback interruption during the video session and the average video quality. These metrics are weighted by three coefficients γ_1, γ_2 and γ_3, which allow us to balance the tradeoff among the QoE metrics according to user preferences.

The weights γ_1 and γ_2 are preceded by a positive sign because the smaller the initial buffering latency and the number of starvation are, the better the QoE is. Implicit tradeoff exists between the two metrics and the initial buffering time is preferred to the starvation by around 90% of users [24]. Although users have a very low tolerance for playback interruptions [23] they also only accept a start-up delay between 5 and 15 seconds, depending on the duration of the overall video [24].

The weight γ_3 is preceded by a negative sign because the higher the average quality, the better the QoE. According to this reasoning, we propose the cost function

$$C(x, N) = \gamma_1 \cdot E[T_x] + \gamma_2 . P_s(j) - \gamma_3 \cdot \sum_{i=1}^{r} w_i T_i \qquad (19)$$

where $E[T_x]$ is the expected initial buffering delay, $P_s(j)$ is the number of starvations, r is the number of available bitrates, w_i is a weight associated to each bitrate and T_r is the fraction of time spent in each bitrate level. We use this function to evaluate the BSC scheme performance on the QoE metrics, i.e., how our scheme can evaluate the objective QoE cost $C(x, N)$. The examples are shown in section V-B.

V. Simulation and Numerical Examples

A. Simulation and validation

The mathematical models of QoE metrics are computed and compared with event-driven simulations using MATLAB. A timer generates random variable stamps that record the frames arrival and departure. Each frame contains two blocks (original frame and the copy of another frame). We monitor the playout buffer length based on the frames arrival rate and the playback rate. A switching occurs when there is no more frames of high bitrate. In this case, the player display only the frames with the low bitrate until high bitrate frames

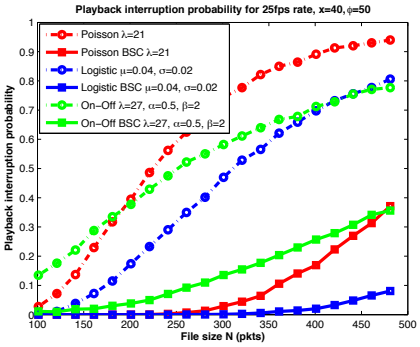


Fig. 11: The playback interruption probability using several packet arrivals processes

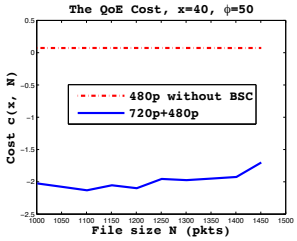


Fig. 12: The QoE cost function for 480p and 720p+480p, $\gamma_1 = 0.1$, $\gamma_2 = 1$, $\gamma_3 = 0.01$

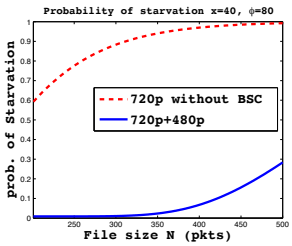


Fig. 13: Reduce the probability of starvation with BSC scheme

are available. A starvation happens when the buffer is empty. We take into account the size of the video file. We vary the parameters frame arrival rate λ, service rate μ, the file size N, the pre-fetching threshold x, the offset ϕ. We run each set of simulations for 4000 times and we show here only a small set of figures obtained, due to page number constraints. Our model exhibits excellent accuracy with the simulations. Figure 9 shows the probability of starvation given the file size N, for different settings of the traffic intensity ρ. The start-up threshold $x = 40$ while the offset $\phi = 50$. The probability of starvation decreases when the network throughput increases. The streaming users only suffer from the playback interruption when $\rho < 1$. We further evaluate the probabilities of having no starvation, at least one and two starvations, given the file size N for $\rho = 0.66$. Figure 10 shows that our analytical model predicts the starvation probabilities accurately. The probability of no starvation decreases from 1 to 0, while the probability of having at least one and two starvations increases. The lag between the two curves gives an idea about the mean playback time, i.e., the mean time between two consecutive starvations. Then, based on the file size, the initial buffering latency, the offset and the traffic intensity, the model predicts the number of starvation that the video session could have.

Before studying the QoE optimization problem, we evaluate the BSC scheme under different types of arrival processes: the logistic distribution and the ON/OFF arrival process. In Fig. 11 we show that the BSC scheme reduces the playback interruption. Furthermore, we obtain an important improvement of the probability of playback interruption where the arrival of frames follows the logistic process.

B. QoE Optimization

We consider five video resolutions (1080p, 720p, 480p, 360p and 240p) with the corresponding bitrates (4500Kbps, 2500Kbps, 1000Kbps, 750Kbps and 400Kbps). The network throughput is 2200Kbps or 3000Kbps. Then we compute the corresponding traffic intensities ρ. We also compute the bitrate in the BSC scheme using H.264/SVC codec. We compare the QoE metrics between a classical DASH based SVC and the BSC system. Our BSC system is based on the SVC codec, then, we compare it to DASH/SVC since we know that SVC adds 10% encoding overhead compared with the same quality AVC. This comparison is done on a single video segment. The bitrate adaptation in the BSC system is the scope of our ongoing research. We will show the two appropriate bitrates to select after the downloading of each segment, and compare the video quality to the classical DASH system.

Let assume that the network throughput is 2200Kbps. In DASH, the adaptation engine will select the bitrate that is just under the network throughput, i.e., 480p resolution. What happens if we select 480p+360p, 720p+360p or 720p+480p in the BSC scheme?

When we select 480p+360p in the BSC, the cost of DASH is better than BSC although there is no starvation in both cases. Indeed, the DASH system just benefits from the initial buffering latency. However, selecting the resolutions 720p+360p or 720p+480p allows to minimize the QoE cost function (Fig. 12). There is only one starvation for a file size of 1500 frames but with a better rendering quality. Hence, when the network throughput changes to 3000Kbps, the BSC can use 1080p video resolution while the simple DASH cannot.

VI. Conclusion and Discussion

In this paper, we proposed the novel Backward-Shifted Coding (BSC) scheme to improve the performance of HTTP adaptive streaming. Inspired from Forward Error Correction, BSC adds time-shifted redundancy to the video stream that provides smooth playback even if the main stream is interrupted.

We describe an integration of BSC into SVC and its operation with HAS. We then provide an analytical characterization of the dominating QoE factors initial buffering time, starvation and average video bitrate. We compute the first two metrics using the Ballot theorem and obtain explicit results for the probability generation function of the starvation. The average video bitrate metric is computed using the quasi-stationary approach. Finally, we show by global optimization that BSC can improve both the playback interruption and the average bitrate in video streaming systems compared to standard DASH on a single video segment.

We will explore the bitrate adaptation in the BSC system in our future work.

Acknowledgment

This work has been carried out in the framework of IDEFIX project, funded by the ANR under the contract number ANR-13-INFR-0006.

REFERENCES

[1] Cisco Visual Networking Index: Global mobile data traffic forecast update, 2014–2019, white paper, Cisco Systems, February 2015.

[2] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica and H. Zhang, *A Quest for an Internet Video Quality-of-experience Metric*, Proceedings of the 11th ACM Workshop on Hot Topics in Networks, Redmond, Washington, 2012.

[3] S. Thakolsri, S. Khan, E. Steinbach, and W. Kellerer, QoE-Driven Cross-Layer Optimization for High Speed Downlink Packet Access, Journal of Communications, vol. 4, no. 9, pp. 669680, Oct. 2009.

[4] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, P. Mogensen, and J. M. Lopez-Soler, QoE oriented cross-layer design of a resource allocation algorithm in beyond 3G systems, Computer Communications, vol. 33, no. 5, 2010.

[5] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica and H. Zhang, Developing a Predictive Model of Quality of Experience for Internet Video, Sigcomm, Hong Kong, 2013.

[6] R. J. Kryscio, C. Lefvre, "On the extinction of the SIS stochastic logistic epidemic, Journal Appl. Prob. 27, 685-694, 1989.

[7] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, Understanding the impact of video quality on user engagement, in ACM SIGCOMM, pp. 362373, 2011

[8] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, Jeffrey Pang, Jia Wang Understanding the Impact of Network Dynamics on Mobile Video User Engagement SIGMETRICS14, June 1620, 2014, Austin, Texas, USA.

[9] C. Yim and A. C. Bovik, Evaluation of temporal variation of video quality in packet loss networks, Signal Processing: Image Communication, Jan. 2011.

[10] ISO/IEC, "Dynamic adaptive streaming over HTTP (DASH)", *International Standard DIS 23009-1.2*, 2012.

[11] FELLER, William. An Introduction to Probability Theory and Its Applications. Volume I. London-New York-Sydney-Toronto : John Wiley & Sons, 1968.

[12] J. Ohm, G. Sullivan, H. Schwarz, T.K. Tan and T. Wiegand, *Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC)*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1669-1684, 2012.

[13] International Standards Organisation/International Electrotechnical Commission (ISO/IEC), *14496-10:2012 Information Technology - Coding of Audio-visual Objects - Part 10: Advanced Video Coding*, 2012.

[14] YE, Zakaria, JIMENEZ, Tania, et EL-AZOUZI, Rachid. Video streaming analysis in Vienna LTE system level simulator. In : Proceedings of the 8th International Conference on Simulation Tools and Techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015. p. 47-54.

[15] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM), 2014.

[16] C. Zhou, C.-W. Lin, X. Zhang, and Z. Guo, A control-theoretic approach to rate adaption for DASH over multiple content distribution servers, IEEE Trans. Circuits Syst. Video Technol., vol. 24, no. 4, Apr. 2014.

[17] C. Liu, I. Bouazizi, and M. Gabbouj, Rate adaptation for adaptive HTTP streaming, in Proc. ACM Multimedia Syst. 2011.

[18] El Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, E. Steinbach, Quality-of-Experience driven Adaptive HTTP Media Delivery, IEEE ICC 2011, Budapest, Hungary, Juni 2013.

[19] C. Lottermann, A. Machado, D. Schroeder, Y. Peng, E. Steinbach, Bit Rate Estimation for H.264/AVC Video Encoding based on Temporal and Spatial Activities, IEEE ICIP 2014, Paris, France, Oktober 2014.

[20] A. Sobhani, A. Yassine and S. Shirmohammadi, "A fuzzy-based rate adaptation controller for DASH", in the Proceeding NOSSDAV 15, 2015.

[21] Parandeh Gheibi et al, Avoiding Interruptions a QoE Reliability Function for Streaming Media Applications, IEEE J. Sel. Areas Commun., Vol.29, No.5, pp:1064-1074, 2011.

[22] XU, Yuedong, ALTMAN, Eitan, EL-AZOUZI, Rachid, et al. Analysis of buffer starvation with application to objective qoe optimization of streaming services. Multimedia, IEEE Transactions on, 2014, vol. 16, no 3, p. 813-827.

[23] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner and P. Tran-Gia, *Quantification of YouTube QoE via Crowdsourcing*, MQoE 2011, Dana Point, CA, USA, (December 2011).

[24] P. Balaouras and I. Stavrakakis, *Multimedia Transport Protocols for Wireless Networks*, Emerging Wireless Multimedia: Services and Technologies (2005), pp. 49-82, 2005.

[25] J.N. Darroch and E. Seneta, *On Quasi-Stationary Distributions in Absorbing Continuous-Time Finite Markov Chains*, Journal of Applied Probability Vol. 4 No. 1, pp:192-196 1967.

[26] I. NÅSELL, *Extinction and quasi-stationarity in the Verhulst logistic model: with derivations of mathematical results*, 2007.

[27] S. Karlin and H. M. Taylor, *A first course in stochastic processes*, Academic Press, New York, 1975.

Sacrificing Efficiency for Quality of Experience: YouTube's Redundant Traffic Behavior

Christian Sieber*, Poul Heegaard†, Tobias Hoßfeld‡, Wolfgang Kellerer*
*Chair of Communication Networks, Technical University of Munich, Germany
{c.sieber, wolfgang.kellerer}@tum.de
†Norwegian University of Science and Technology, Trondheim, Norway
{poul.heegaard}@item.ntnu.no
‡Universität Duisburg-Essen, Germany
{tobias.hossfeld}@uni-due.de

***Abstract*—Internet traffic reports show that YouTube is one of the major sources of data traffic world-wide. Furthermore, the data traffic shifts from mostly fixed landlines to cellular data connections where bandwidth is sparse and expensive. Previous studies revealed that YouTube uses a user-friendly HTTP Adaptive Streaming (HAS) strategy which sacrifices bandwidth efficiency to increase the average playback quality for the user. That way, it happens that the same video segment is transmitted in two or more quality levels, but only one can be shown to the user. We denote this as redundant traffic and this work is dedicated to understanding the influence factors on the amount of redundant traffic. This paper presents the results of a large-scale study with over 12,000 video views over a bottleneck link shaped to various bandwidths.We first evaluate the playback characteristics and show that YouTube's HAS algorithm linearly increases the average playback quality with the available bandwidth while at the same time video buffering is sub-linearly decreased. Furthermore, we identify video-dependent bandwidths which optimize the playback time on a quality level. Afterward, we show that this is achieved by discarding lower layer segments and therefore paid with redundant traffic of up to 40 %. We evaluate the overall efficiency of the system and show that YouTube is able to improve the average quality level by up to 0.7 quality levels by using this adaptation strategy. However, a penalty of 0.5 quality levels is paid for it due to the discarded data of the lower quality segments.**

I. Introduction

Video streaming services in the Internet gain more and more importance. The offered content and services for video on-demand (VoD) including live streaming exhibit a strong increase in popularity. By now, Internet traffic reports account VoD for the largest single type of consumer traffic in the Internet for landline and cellular access [1].

HTTP over TCP has become the de-facto standard for delivery of the VoD content. The HTTP protocol is firewall-friendly and easy to implement. Furthermore, large content delivery networks (CDN) are in place to distribute HTTP content globally and provide the content close to the users. At the beginning, VoD over HTTP was implemented by progressive download [2]. With progressive download, a HTTP server provides a single file with the content in a specific quality level. The player, e.g., the browser, starts to download the file and at the same time, or after an initial buffering phase, begins to play the video to the user. However, in case of insufficient network bandwidth, the video buffer depletes and the video stalls, which significantly decreases the Quality of Experience (QoE) of the end user [3].

ISBN 978-3-901882-83-8

Progressive download does not allow to adapt to current network conditions or to specific devices, e.g., phone or TV screen. By now, progressive download is being replaced by HTTP Adaptive Streaming (HAS). See [2] for a survey on HAS. HAS encodes the content into different quality levels, segments it into small chunks of a few seconds and makes it available over HTTP. A manifest file describing the content, e.g., video codec, number of quality levels and chunk location, is placed alongside the segments on the HTTP server. At first the client requests the manifest file and afterward downloads the content segments in a quality level chosen by the client. Dynamic Adaptive Streaming of HTTP (DASH) is an ISO standard describing the structure of the manifest file and is in use by some of the major content providers. However, the quality level adaptation logic, which dictates how the client should adapt the quality, is not part of the standard and is left to be decided by the implementation of the streaming client. Factors to consider for the adaptation are for example the viewing device, the available download bandwidth and the video bit-rates of the quality levels. As every client can implement their own adaption algorithm, the QoE of the user differs between content providers and streaming clients. The QoE of adaptive streaming is an active research topic [2]. QoE studies show that stalling must be avoided [3], [2] and that the quality switching rate should be minimized [4]. The average quality level is a major influence factor for the QoE in adaptive streaming [4].

In this paper we take a closer look at one of the major sources of video streaming traffic [5] in the Internet, YouTube. In a first pilot study [6] we have shown that YouTube's adaptation algorithm replaces previously downloaded lower quality segments. However, this introduces overhead, denoted as redundant traffic, as the lower quality segments are discarded. Hence, there is a trade-off between network efficiency and average playback quality that also affects the Quality of Experience of the user. Due to the limited scope of the study (four videos in total, 150 views), no reliable conclusions could

be drawn with respect to possible influence factors.

In this paper, we tackle two important questions. First, when does redundant traffic occur and how much redundant traffic is transferred over the network? Second, what is the overall efficiency of this approach, i.e., how much of the additional traffic can actually be used to improve the average quality level and how much has to be paid as penalty for replacing lower layer segments? To answer this, over 12,000 video views were recorded in our test-bed under different emulated network bottlenecks. We first describe the playback characteristics as observed for the different network conditions from the end user's point of view. Afterward we relate the playback characteristics to the recorded redundant traffic and show that the amount of redundant traffic can be estimated based on the playback characteristics. At the end, we evaluate the overall efficiency by comparing the quality gain due to the adaptation strategy with an estimation of the maximum achievable quality level based on the amount of total downloaded data in the session.

This paper is structured as follows. We first describe the background and related work in Section II. Here we also explain the behavior which leads to the redundant traffic in detail. In Section III we introduce the measurement methodology and explain the measurement set-up. In Section IV we discuss general playback characteristics observed in the study. In Section V we first evaluate the relationship between the observed playback characteristics and the overhead introduced by discarding lower quality segments. Afterward we evaluate the overall efficiency of the adaptation strategy considering the overhead. In Section VI we conclude the work and give future research directions.

II. Background & Related Work

In the following we first introduce HTTP adaptive streaming in general. Afterward we discuss the results of our previous study where we do a first characterization of the adaption behavior of YouTube. At the end we revisit the related research to this topic.

A. HTTP Adaptive Streaming

HAS enables client-driven adaptation of the quality level to device capabilities such as screen size, resolution and stereoscopic capabilities. Furthermore, by switching to different quality levels, the client can adapt the video bit-rate to the available bandwidth and therefor adapt to dynamically changing network conditions such as observed for example in cellular environments. HAS is implemented by encoding the content into multiple representations, e.g. different quality levels, segmenting it in small chunks (for YouTube: 10 s to 20 s, depending on the video) and making the segments available to the client through the HTTP protocol. The ISO standard MPEG-DASH is a widely accepted HAS standard adopted by YouTube. DASH defines an XML-based Media Presentation Presentation (MPD) file which describes the representations, e.g., average bit-rate or codec used, and gives the URL to the individual segments. At the beginning of a streaming session, the client requests the MPD file. Afterward the implementation on the client-side decides which chunks to request from which representation. As every content provider can implement their own adaption strategy, the experience for the user depends not only on the offered representations, but also on the ability of the adaption strategy to request in chunks in a user-friendly way. Evaluations show that the adaptation algorithm has a strong influence on the resulting playback behavior and therefore also on the perceived QoE of the user [7]. It has to be noted that YouTube also allows the end user to manually select a fixed quality level which deactivates HAS even if it results in frequent buffering. However, the default setting is the automated quality adaptation.

B. YouTube's Quality Adaptation Strategy

In our previous study [6] we describe the behavior of YouTube where the streaming client replaces previously downloaded chunks with higher quality ones. Figure 1 illustrates a request schedule for one of the experiment runs. At first, the YouTube player requests four segments of quality level 144p of a playback time up one minute and all four requests are made in the first 20 seconds of the experiment. At about 21 seconds into the experiment, the player revises its decision and replaces the segment containing the playback time 30 s to 45 s with a quality level of 240p. Afterward the player switches back to 144p and downloads playback time 60 s to 90 s in quality level 144p. Later in the experiment, the player switches up to a quality level of 480p by replacing 360p and 144p segments. The figure illustrates that some chunks of content are downloaded even more than twice.

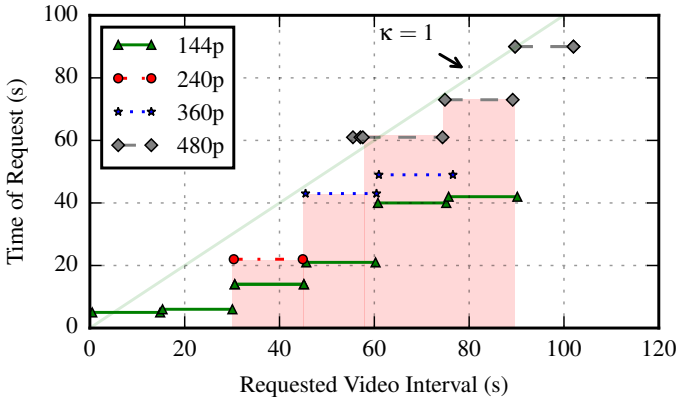


Figure 1. Example request schedule. The shaded areas indicate where lower quality segments are replaced by higher quality segments. For the first minute into the playback, 30 seconds of 144p are replaced with 15 seconds of 240p and 15 seconds of 360p to increase the average playback quality.

From the request schedule it becomes obvious that the adaption strategy tries to optimize the average playback quality shown to the user. Furthermore, by constantly downloading chunks, the algorithm prevents stalling of the underlying TCP connection. In a previous study [7] we show that by constantly utilizing the TCP connection a HAS strategy can keep the fair share of the network bandwidth provided by TCP adaptive behavior. Furthermore, adaptation strategies that underutilize

the TCP connection, e.g., if the playback buffer is full, have a distinct disadvantage compared to strategies which constantly utilize the TCP connection.

C. Related Work

In [8], Añorga et al. present a recent study about YouTube's HAS adaptation and review previous studies. Their findings show that YouTube uses a large playback buffer and therefore reacts only slowly to changing bandwidth conditions (13 s to 40 s). In [9], Yao et al. evaluate different sources of redundant traffic in video streaming services. They show how the iOS YouTube player requests overlapping segments to smoothen the playback. The amount of redundant traffic is not evaluated. In [10], Lui et al. evaluate the difference between Android and iOS-based YouTube media streaming. They conclude that buffering is not dependent on the playback time, but on the amount of data buffered. Furthermore, they quantify a redundant traffic of 15 % and account it for re-downloading the beginning of the video. Mansy et al. [11] analyze the streaming behavior of the three streaming providers including YouTube in terms of their playback characteristics, redundant traffic and bandwidth utilization. The authors observe that YouTube aggressively discards segments of lower quality levels to download higher bit-rate segments when the bandwidth increases. In the study, one video is evaluated in a wireless scenario with varying bandwidths (every 2 minutes a new link bandwidth is set). For this scenario, the authors conclude a percentage of redundant traffic of 16 %. The study also shows that other content providers deploy similar adaptation strategies. Nam et al. [12] show that in a mobile scenario more than 35 % of transferred data by YouTube is redundant. The authors account frequent termination of TCP connections and discarded on-fly packets as the case of the redundant traffic. Rao et al. [13] and Ito et al. [14] model the traffic patterns produced by a YouTube streaming session. Their findings suggest that the implementation of the adaptation strategy and therefor the resulting behavior varies with the type of the viewing device. Alcock et al. [15] describe the initial burst phase deployed by YouTube. In the initial burst phase, 32 s of playback time are sent to the client a fast as possible. We also observe this initial burst phase and account it for a source for redundant traffic, as a low quality initial burst phase is in some cases later replaced by higher quality segments.

This work extends the state-of-the art and presents the first large-scale study which statistically quantifies the amount of redundant traffic. Further, we relate redundant traffic to QoE influence factors and are able to quantify the QoE loss by downloading redundant traffic instead of higher quality levels.

III. Methodology

In the following we first discuss the experimental set-up used to collect the results. Afterward we introduce the notation and metrics evaluated in this work. At the end we give a description of how we selected the content and which characteristics the selected content exhibits.

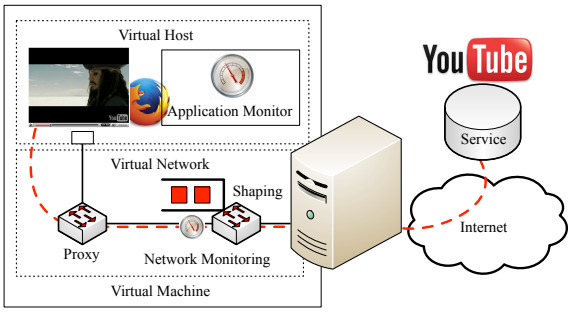


Figure 2. Experimental set-up based on a virtual machine (VM) and virtual network with browser-based and in-network-based monitoring.

A. Experiment Set-up

Figure 2 illustrates the experiment set-up. The set-up consists of a virtual machine with *Xubuntu 14.04 64-bit* running a browser (*Firefox 21*) with the YouTube player, an HTTPS proxy inside the virtual machine and a virtual network which limits the available bandwidth. The set-up is connected to the Internet through a lightly utilized lab network and through the university's Internet connection. Internal traffic reports and monitoring of experiment download speed rule out external bottlenecks. Since the beginning of 2015, YouTube uses the encrypted HTTPS protocol for video delivery. In order to still being able to decrypt the traffic in the network, we inject a custom certificate into the browser and mark it as trustworthy for the domains used by YouTube. Furthermore, we use a customized version of *mitmproxy* [16] in order to intercept, capture and decrypt the YouTube traffic. The proxy was customized to quickly forward all incoming traffic without buffering and tested to rule out any performance issue as influence factor on the adaptation behavior.

In order to capture the player state and the HTTP requests in the network, we use browser-based and network-based monitoring. We implemented browser-based monitoring by embedding the YouTube video into a custom web-page and use an extension for the browser [17] to access the API of the player to monitor buffering events and quality switches. For network monitoring, we use the described proxy to capture and decrypt the HTTPS requests. In order to translate between the byte range requests and playtime seconds, we download the videos [18] in all evaluated quality levels and afterward decode the downloaded mp4 containers [19].

Before every experiment run, the virtual environment is reset to a default state. During the playback, the status of the experiment is constantly monitored and if the video was not played out until the end, the experiment run is discarded. A detailed description including the whole experimental set-up is available together with the raw and summarized traces online [20]. We encourage others to play with the data and do their own evaluations. At total, the traces and summarized results for 12,075 runs are available online (35 videos $\times$ 27 bandwidth values $\times$ 15 replications).

Table I
KEY VARIABLES AND NOTATIONS USED IN THE PAPER.

notation	meaning
B	downloaded and played bytes
B_T	total downloaded bytes
ρ	Redundant Traffic Ratio (RTR)
Q	number of quality levels
$\mathbb{Q}$	set of quality levels; $\mathbb{Q} = \{q_0; q_{Q-1}\}$
q	quality level, $q \in \mathbb{Q}$
V	number of videos
v	video index
R	number of replicated downloads (of each video v)
r	replication index
F	number of bandwidth levels
f	bandwidth level
f^*	rescaled bandwidth level
n_v	number of segments in video v
τ_v	(fixed) segment play time of video v
x_{ij}	quality index indicator; 1 if segment i is downloaded at quality level j, 0 otherwise
s_{ij}	size of segment i for video quality level j
J	average quality level
J_i^+	maximum quality level downloaded of segment i; $J_i^+ = \{\max j \| x_{ij} > 0\}$
J_i^-	minimum quality level downloaded of segment i; $J_i^- = \{\min j \| x_{ij} > 0\}$
b_i	number of buffer event in segment i
ψ	buffering rate for bandwidth f
t_{fv}	buffering time for video v and bandwidth f
T_{fq}	average relative time/probability on quality level q in a video sequence with bandwidth f
γ	average bit-rate
ϕ_i	1 if quality level has switched from segment $i-1$ to i, 0 otherwise;
w	average number of switches
ϵ	overall efficiency
κ	buffering ratio

B. Metrics

The notation used in the paper is summarized in Table I. In the following we define the key performance metrics as used in our analysis. Efficiency ϵ is defined in Chapter V.

The average bit-rate per quality level q of video v

$$\gamma_{qv} = \frac{1}{n_v \tau} \sum_{i=1}^{n_v} \{s_{iq}\}_v \tag{1}$$

Total downloaded bytes in video sequence v:

$$B_T = \sum_{i=1}^{n_v} \sum_{j=0}^{Q-1} x_{ij} s_{ij} \tag{2}$$

Total *played* bytes in a session under the assumption that always the maximum quality level downloaded J^+ is played:

$$B = \sum_{i=1}^{n_v} s_{iJ_i^+} \tag{3}$$

Each bandwidth level f, replication r and video v has a unique B_{frv} and a corresponding average quality level J_{frv}. The average quality level with played bytes B is denoted J_B and is the average over the video sequences with $B_{frv} = B$.

The average played quality level for bandwidth f:

$$J_f = \frac{1}{RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{J_i^+\}_{frv} \tag{4}$$

The average buffering event rate for bandwidth f:

$$\psi_f = \frac{1}{\tau RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{b_i\}_{frv} \tag{5}$$

The average quality switching rate for bandwidth f:

$$w_f = \frac{1}{\tau RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{\phi_i\}_{frv} \tag{6}$$

The estimated probability of playtime on quality level q in a video sequence for bandwidth f:

$$T_{fq} = \frac{1}{RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{x_{iq}\}_{frv} \tag{7}$$

The buffering ratio κ (8) is the ratio between the experiment run-time and the duration of the video:

$$\kappa_{frv} = \frac{n_v \tau_v + t_{frv}}{n_v \tau_v} \tag{8}$$

Buffering ratio $\kappa = 1$ means that no buffering happened during the experiment. κ for a specific bandwidth f is defined as $\kappa_f = \frac{1}{RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \kappa_{frv}$.

C. Redundant Traffic Ratio (RTR)

The redundant traffic is defined as the ratio between the downloaded bytes that are discarded ($B_T - B$) and the bytes that are played (B). This gives the overhead of the playback with the data volume of only the played out segments as reference. The Redundant Traffic Ratio (RTR) for one single download is then

$$\rho = \frac{B_T - B}{B} \tag{9}$$

For download at a specific bandwidth factor f, the index is added to B_T in Eq. (2) and to B in Eq (3) and the redundant traffic ratio ρ is updated accordingly.

D. Content

In order to represent the variety of videos uploaded to YouTube, we use the YouTube API provided by Google to automatically select suitable videos. We define 5 categories "*minecraft*", "*music*", "*funny cats*", "*gopro*", "*game*" by using the category name as search query string. Furthermore, we filter the query results by the following criteria. The selection of videos considers the following aspects: 1) embeddable, i.e., use in HTML iframe allowed, 2) syndication is allowed, 3) available in high resolutions and 4) videos which were published between 1 and 9 month ago. The query result is sorted by popularity, i.e., view count, and from the result we select videos with a duration of 1, 2, ... , 10 minutes with an allowed deviation of 5 seconds. In total, 35 different videos were accessible during the whole experiment time and are

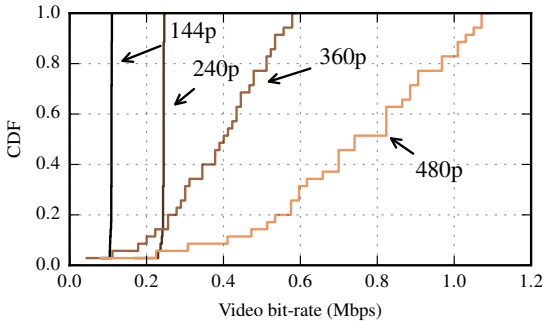


Figure 3. CDF of the quality levels of the selected video sequences. Means: 144p: 0.11 Mbit/s, 240p: 0.24 Mbit/s, 360p: 0.37 Mbit/s, 480p: 0.72 Mbit/s.

included in the evaluation. Average duration of selected videos is 5.3 minutes.

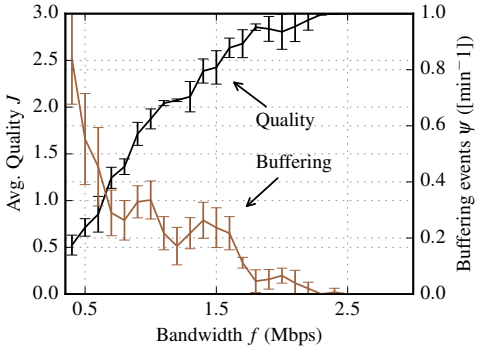
Figure 3 shows the cumulative distribution function (CDF) of the bit-rates of the four quality levels, $\mathbb{Q} = \{144p, 240p, 360p, 480p\}$ for the selected videos. Note that the client player uses some decision logic to select a subset of the available resolutions on YouTube's servers based on the type of device, e.g. based on the screen size. In our environment this was the subset $\mathbb{Q}$. The average bit-rate for each quality level is calculated by Eq. 1, then γ_{144p}=0.11 Mbit/s, γ_{240p}=0.24 Mbit/s, γ_{360p}=0.37 Mbit/s, and γ_{480p}=0.72 Mbit/s. Note that although 360p has a higher number of pixels than 240p, approximately 18 % of the videos in quality level 360p are encoded with a lower average bit-rate then the average bit-rate for quality level 240p. YouTubes encoding of (cover art) music videos leads to higher data volume for lower resolutions than for higher resolutions. The reason for this is not clear as it depends on the YouTube internal encoding of those videos. Please note that we nevertheless include those videos in the first part of the evaluation as they are part of the content mix observed on the platform. For the evaluation of the efficiency, we exclude those videos.

IV. Video Playout and Adaptation Characteristics

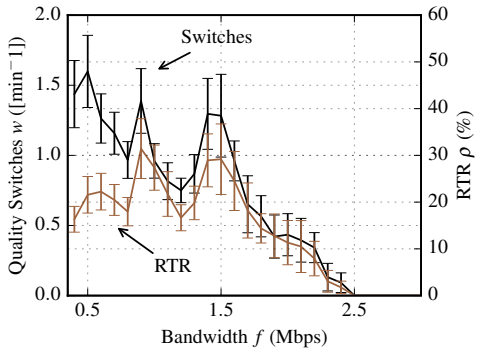
This section presents key characteristics of the adaptation observed in our measurements for different bandwidths f. This includes the average quality level J_f, quality switching rate w_f, buffering rate ψ_f, and the probability of a playback T_{fq} at quality q in a session. If not otherwise stated, error bars in the figures depict the 95 % confidence interval.

Figures 4 (a) and (b) illustrate how the average quality J_f, the buffering rate ψ_f, the quality switching rate w_f and the redundant traffic ratio ρ_f develops for increasing bandwidth f for video v = CbhnuRhbC, which shows scenes from a video game with picture-in-picture commentary. The average bit-rate for the four quality levels are γ ={0.11 Mbps, 0.25 Mbps, 0.43 Mbps, 0.84 Mbps}.

Figure 4(a) shows the average quality J and the buffering rate ψ. The buffering rate decrease from 0.8 [min^{-1}] at 0.8 Mbps to 0 at 2.5 Mbps. The buffering rate has two peaks for ρ. The average quality level increases approximately linear from 0.5 at 0.8 Mbps to the maximum of 3 at 2.5 Mbps.



(a) Buffering rate and average quality



(b) Quality switching rate and RTR

Figure 4. Average playback quality J_f, buffering rate ψ_f [min^{-1}], quality switch rate w_f [min^{-1}] and RTR ρ_f for video v = CbhnuRhbX5I. 95 % confidence intervals over 15 runs are indicated.

In Figure 4(b) we observe that for bandwidth $f \leq 1.6$ Mbps, the RTR is approximately $\rho = 20$ %, except for two peaks of 30 % at both 0.9 Mbps and 1.5 Mbps. With $f >$1.6 Mbps, the RTR decreases linearly until it reaches zero at about 2.5 Mbps. The quality switching rate has a linear decreasing trend from approximately 1.5 [min^{-1}] down to 0 [min^{-1}] at 2.5 Mbps. The same two peaks at 0.9 Mbps and 1.5 Mbps as for the RTR ρ_f are also observed for w_r. Next we summarize the results over all videos per bandwidth f.

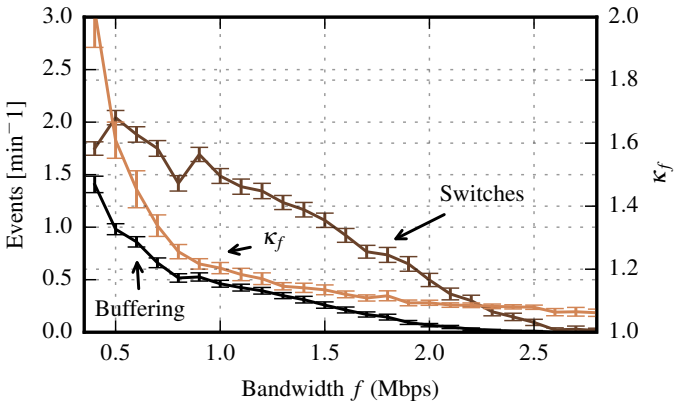


Figure 5. Buffering events [min^{-1}], switching events [min^{-1}] and buffering ratio κ_f. κ_f and buffering rate decreases non-linear. Switching rate decreases linear.

Figure 5 shows the buffering rate, the switching rate and the

average *buffering ratio* κ for the different bandwidth values f and averaged over all videos. The buffering ratio κ (8) is the ratio between the experiment run-time and the duration of the video. $\kappa = 1$ means that no buffering happened during the experiment. In our set-up, the browser start-up time is included in the experiment time. The minimum value κ can reach is about 1.05, depending on the video. The figures show that the buffering rate and average buffering ratio decreases non-linear with increasing f, while the corresponding decrease in quality switching rate is approximately linear. At the lowest evaluated bandwidth 0.4 Mbps, we observe an average buffering rate of 1.4 [min^{-1}] and an average of buffering time close to two times the duration of the video. The quality switching rate is on average between 1.75 [min^{-1}] and 2.0 [min^{-1}] for 0.4 Mbps to 0.5 Mbps. At about 2.6 Mbps, the three metrics reach their minimum of zero for the buffering and switching events and about 1.07 for the ratio between experiment and video duration. From the figure we conclude that for a bandwidth of 2.6 Mbps all videos in our result set are, on average, played back without buffering events and quality switches. Furthermore we see that switching events are more frequent than buffering events and decrease slower for increasing f.

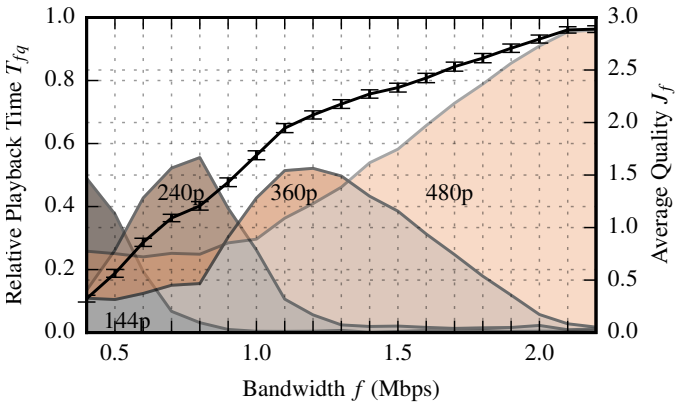


Figure 6. Average quality level and probability of playout time on quality level T_{fq} ($\sum_{q \in \Omega_I} T_{fq} = 1$).

In Figure 6, the average quality level is depicted on the axis to the right. The axis on the bottom gives the bottleneck bandwidth in the range of $f = (0.4, 2.2)$ Mbps. The shaded areas indicate the average fraction of time spent on each of the quality levels for a specific bandwidth, i.e., the relative playback time (scale on the left axis). Two key observations can be made from the figure. First, for each quality level there is a distinct bandwidth range where one of the four different quality levels dominates. This is illustrated by T_{fq} for $q = 0, 1, 2, 3$ which have their maximum at different bandwidths f. Second, even at a low data rate as $f = 0.5$, the $T_{0.5,4} = 0.2$, which means that in 20 % of the time the highest quality level is viewed to the user. This can be explained by the fact that some of the videos, especially music videos with static cover, have a low average bit-rate for the higher quality levels and therefor can be selected in the highest quality level even if the available download-rate is low. The figures also illustrate for which range of f the different qualities dominate and when we switch from one level to the next higher or lower quality level. Based on Figure 6 we cannot identify a clear relationship between the bandwidth and the probability of selecting a specific quality level as the regions are heavily overlapping. This is due to the high variance in video bit-rates as shown in Figure 3.

To align the probabilities of the four quality levels, T_{vq}, we scale the bandwidth f with a quality and video dependent bandwidth factor $f^*_{vq} = f/\gamma_{vq}$. In Figure 7 we plot the relative playback time T_{vq} for each experiment run and for all four quality levels q using the rescaled bandwidth f^*_{vq}. The error bars indicate the confidence interval of 95 % for each of the bins. From this we observe that $q_1 =$ 240p, $q_2 =$ 360p and $q_3 =$ 480p reach their maximum T_{fvq} at approximately $f^*_q = 3$. This is indicated by the vertical (red) line. For 240p and 360p the maximums are close to 60 %, while for 480p the maximum is 100 %. For 144p, the maximum of about 55 % is reached at $f^*_1 = 3.8$. There are no samples for $f^*_1 < 3.8$ available, which corresponds approximately to a bandwidth of 0.4 Mbps. Hence, the maximum is reached for all quality levels at $f^* = 3$, independent of the quality level q. It can be read as; when three times the average bit-rate of a certain video v of a certain quality level q is available, this quality level dominates. If we have more, we switch to a higher level, if it exists. Note that we compare here the link bandwidth with the raw video bit-rate, without any overheads (e.g. IP, HTTP, TCP or video container overhead) and without the audio stream. Therefor, the absolute value where T_{vq} reaches its maximum may vary depending on the scenario. But, the results show that there is a (narrow) range of bandwidths for each video where each quality level reaches a maximum probability of being played out to the user. This suggests, that the adaptation algorithm of YouTube uses the average bit-rate of a quality level of a video and compares it with the available network bandwidth to determine which quality level to select next. The results also reveal that although the network bandwidth remains constant quality switches occur and the video is watched in different quality levels. In the next chapter we discuss how playback characteristics affect on the redundant traffic.

V. Evaluation of Adaptation Efficiency

In the previous section we characterized the playback behavior from the perspective of the user (average quality level, buffering and switching events) and discussed how the available bandwidth influences the playback behavior. Next, we put the evaluated playback characteristics in perspective to the amount of redundant traffic downloaded in the background, unnoticed by the viewer, and subsequently discuss the effectiveness of this adaptation approach. We do this by introducing a metric which summarizes the quality gain due to the adaptation on the one side and the penalty introduced by the redundant traffic on the other side.

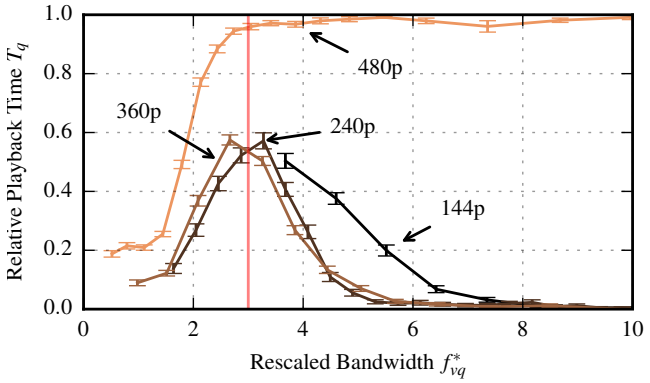
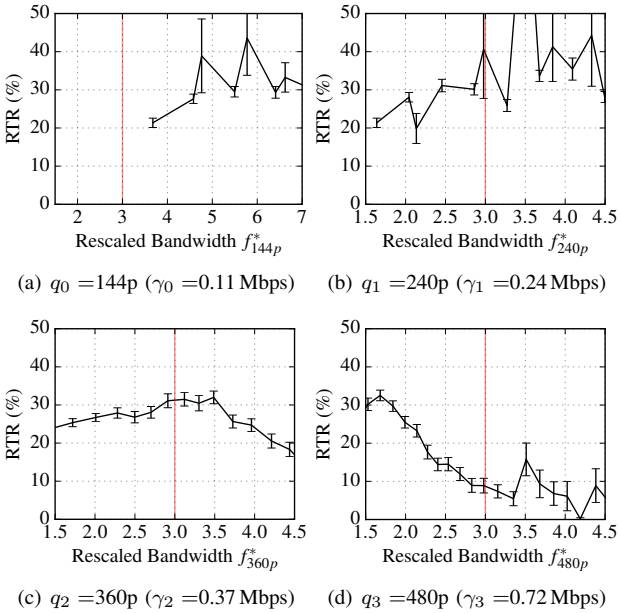


Figure 7. The probability of each playback quality level $T_{f^*_{vq}}$ as a function of the rescaled bandwidth f^*_{vq}. The vertical line at $f^*_{vq} = 3$ indicates where we find the peak for all quality levels.

A. Relationship of Playback Characteristics to Redundancy

In Figure 8 we take a closer look at the peak of relative playback time as shown in Figure 7 from the perspective of the RTR based on the rescaled bandwidth f^*_{vq}. The average bit-rate of each of the quality levels q is given as a reference and can be used to get an estimate of the bandwidth f based on f^*_q ($f = \gamma_q \times f^*_q$). However, this neglects the standard deviation of the bit-rates of the videos. The figure shows that for 240p and 360p, high T_q translates to about 30 % of RTR. For 144p, a RTR of 20 % for 3.6 of rescaled bandwidth is observed. The highest quality level 480p exhibits a RTR of 10 % at the point where it reaches close to 100 % of the playback time. We conclude from the figure that the peak of the estimated probability of playtime on a quality level q, T_{fq}, does not translate to a stable, i.e., sequential, quality selection behavior. On the contrary, the RTR for 240p and 360p show that the player downloaded up to 40 % of data more than it showed to the user.

Figure 9 depicts the switching and buffering rate for increasing values of RTR. K-means clustering is used to generate bins of RTR in the figure. The error bars indicate the 95 % confidence interval for a cluster. The shaded area in the background shows the probability density function, $g(\rho)$ for the observed values of RTR. The $g(\rho)$ shows that approximately 25 % of the experiment runs in the result set do not exhibit any or a minor percentage of redundant traffic and the second highest density of samples can be observed between a RTR of 15 % and 50 %. The figure illustrates that there is a close relationship between the switching/buffering rate and the amount of redundant traffic. While the buffering rate increases only slowly for increasing RTR, the switching rate increase is steeper. For 20 % of RTR the buffering rate is approximately one buffering event per two minutes, while switching rate is up to 1.2 switches per one minute. By implication, this shows that buffering events quickly translate to a high amount of redundant traffic. Buffering events are a sign of sudden drop in bandwidth (or equivalent: a sudden increase in the video bit-rate) or an effect of poor decision



(a) q_0 =144p (γ_0 =0.11 Mbps) (b) q_1 =240p (γ_1 =0.24 Mbps)

(c) q_2 =360p (γ_2 =0.37 Mbps) (d) q_3 =480p (γ_3 =0.72 Mbps)

Figure 8. RTR for the rescaled bandwidth f^* relative to the four quality levels. Note that the shown range of f^* is wider for 144p compared to the other quality levels.

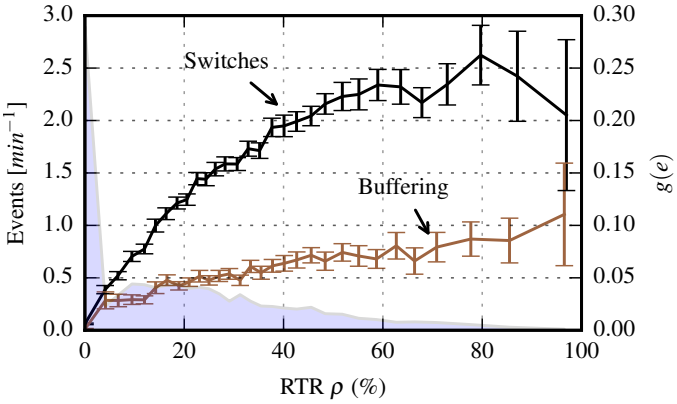
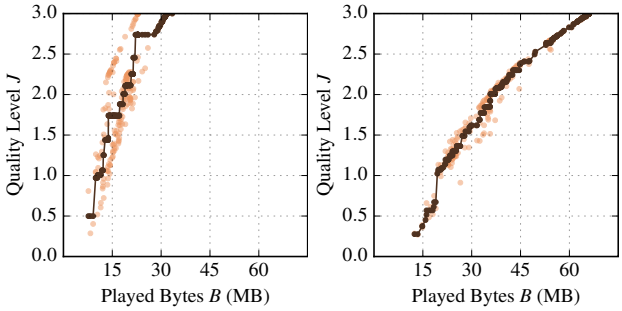


Figure 9. Switching and buffering events for increasing RTR. The shaded area illustrates the probability density function of the RTR values in the collected result set.

by the adaptation logic. Hence, it can be concluded that higher values of redundant traffic can be observed when the playout buffer is often depleted. Furthermore, from the (approximately) strict monotonic increase follows that it is possible to estimate the RTR of a viewing session based on the playback characteristics.

B. Efficiency of Adaptation Strategy

Next, we discuss the efficiency of the observed adaptation strategy. In particular we want to answer the following questions: How much of the additionally downloaded data was actually used to improve the average playback quality? And how much average quality was lost, compared to an optimum where the segments are downloaded without overlaps. For calculation of the efficiency we exclude videos where higher quality levels have a lower bit-rate than lower quality levels.



(a) v_1 = _pjBVUyq530 (b) v_2 = vbLLqaa9ksw

Figure 10. Approximated function θ for two of the videos.

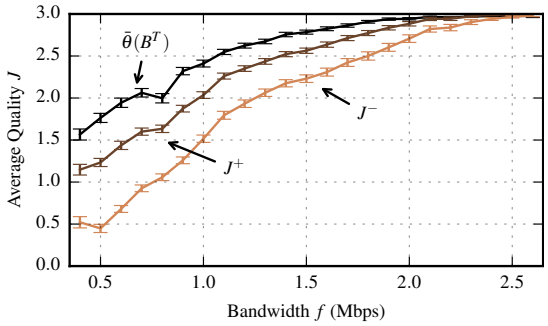


Figure 11. The worst case average quality J_f^-, the played average quality J_f^+ and the estimated optimum quality $\theta_f(B^T)$ for increasing bandwidth f.

The translation between average playback quality of a certain downloaded volume of bytes z is estimated by a function $\theta(z)$, which is determined by regression of the observation of the quality levels as a function of the bandwidth f. We apply isotonic regression [21] to fit the played bytes B to the average quality level. Isotonic regression approximates monotonic functions and does not assume a specific shape of the underlying function. The translation between the bytes B of the *viewed* quality level J^+ and the quality level J^+ itself is (mostly) monotonic, therefor isotonic regression is applicable to the problem at hand.

Figure 10 illustrates the isotonic regression result for two of the videos v_1 and v_2 used in the result set. The red dots depict the samples collected for the video, (B, J^+). The connected green dots show the approximated function $\theta(y)$, $y \in (0, B^+)$ (B^+ is the maximum observed B). Two observations can be made from the figure. First, θ_{v_2} is more flat than θ_{v_1}. Second, v_2 shows a larger deviation for the bytes required for a specific average quality level. The difference in slope is a result of the different bit-rates for quality level q_3 = 480p for the two videos. q_3 = 480p of video v_1 has an average bit-rate of $\beta_{v_1} = 0.49$ Mbps, while for v_2 the average bit-rate is $\beta_{v_2} = 0.78$ Mbps. The deviation for the same quality level indicates that the standard deviation for the video bit-rate for video v_1 is higher than for v_2. Next, we use θ to estimate the optimum average quality level for a given B^T and compare it to the observed average playback quality and to the worst case quality level J^- where no lower quality segments were replaced by segments with a higher quality.

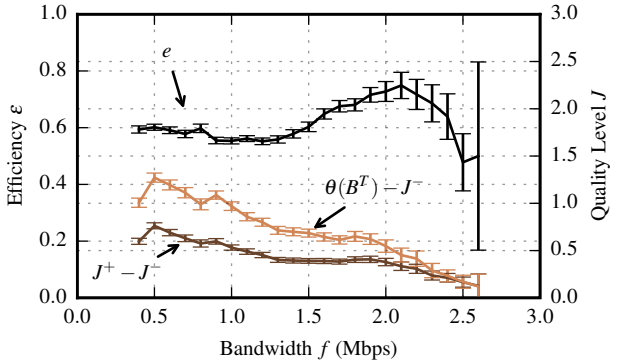


Figure 12. Efficiency ϵ_f, difference between played and worst case quality level $J_f^+ - J_f^-$ and difference between estimated optimum and worst case quality $\theta_f(B^T) - J_f^-$ over bandwidth f.

Figure 11 depicts the worst case quality level J_f^-, the played quality level J_f^+ and the estimated optimum quality level based on the total downloaded amount of data $\theta(B^T)$ for a bandwidth factor up to f = 2.6 Mbps. It can be seen in the figure, that for low bandwidths, the absolute difference between three quality levels is larger than for higher bandwidth values. For example for 0.5 Mbps, the quality level increases due to the adaption strategy by 0.7 quality levels. However, based on the amount of downloaded data, the redundant traffic introduced a penalty of 0.5 quality levels. For higher bandwidth values, the absolute difference between the three quality levels J_f^-, J_f^+ and $\theta(B^T)$ becomes less, but the ratio between them stays roughly the same. For a bandwidth of 2.6 Mbps, no difference can be observed.

Next, we define a metric for the efficiency of the adaptation strategy in respect to the amount of downloaded and played data. In the best case, an adaption strategy allows the player to utilize all the bytes downloaded. However, this is only possible if $\theta(B^T) = J_f^+ = J_f^-$, as each difference between J_f^+ and J_f^- introduces redundant traffic and therefor increases also the difference between estimated optimum $\theta(B^T)$ and J_f^+.

We define the *efficiency* ϵ of the algorithm as the ratio between the measured quality gain $(J^+ - J^-)$ and the estimated maximal quality gain $\theta(B^T) - J^-$ (see Eq (10)).

$$\epsilon = \frac{J^+ - J^-}{\theta(B^T) - J^-} \tag{10}$$

If $\epsilon = 1$, the algorithm was able to utilize all the additionally downloaded data to improve the average quality level. If $\epsilon = 0$, the algorithm requested additional segments, but was not able to improve the average quality level by doing so.

Figure 12 depicts the efficiency ϵ_f, the difference between played and worst case quality level $J_f^+ - J_f^-$ and difference between estimated optimum and worst case quality $\theta_f(B^T) - J_f^-$ over bandwidth f. The figure shows that for low bandwidths ($f < 0.8$ Mbps), the adaptation strategy is able to utilize 60 % of the additionally downloaded data to increase the average quality level J. For bandwidth from 0.8 Mbps to 1.5 Mbps, the efficiency drops to 55 %. For bandwidths from 1.5 Mbps up to

2 Mbps, the efficiency increases up to 65 %. For bandwidths larger than 2 Mbps, the width of the confidence interval do not allow a reliable conclusion. For overcapacity bandwidth f, the ϵ_f looks "noisy" because there are no adaptation, and hence not redundancy, and therefor the $\theta_f(B^T)$, J_f^-, and J_f^+ are (almost) equal. To summarize the figure, the algorithm is on average able to utilize 60 % of the additionally downloaded data to improve the average quality level compared to J^-, where no replacement of lower quality segments takes place. The left over 40 % are the penalty the algorithm has to pay for discarding previously downloaded segments.

VI. Conclusion & Outlook

The traffic share of video streaming in the Internet continues to increase. Major content providers such as YouTube provide a global infrastructure to serve commercial and user-generated contents to every end-user's device. YouTube alone accounts for about 15 % of the total downstream Internet traffic in the US. Furthermore, reports show that the video traffic also increases for cellular access where available data-rate is scarce. Therefor it is important to understand how YouTube adapts to the available bandwidth and how efficiently it uses the available network resources. Previous studies reveal that the adaption strategy used by the YouTube client tries to optimize the average playback quality by replacing previously downloaded lower quality segments with higher quality segments. This increases the average quality shown to user. However, it decreases the efficiency in respect to the used network resources. In this paper we present the results of a large-scale study with more than 12,000 video views of different contents while the downstream traffic was shaped to emulate a bottleneck link with a certain bandwidth. First we show how YouTube adapts to the available bandwidth from the perspective of the user in terms of quality switching, buffering events and playback quality. Higher available bandwidth increases almost linearly the average playback quality, while buffering ratio and buffering event rate are decreasing sub-linearly. The switching frequency is decreasing almost linearly. The results show that for every video and quality level there is a specific network bandwidth which maximizes the time spent on the specific quality level. Thus, this network bandwidth is related to the video bit-rate of that quality level. However, that specific bandwidth does not force the player to select a specific quality level with a probability larger than 0.6 for any quality level below the maximum, except for the trivial case of bandwidth overprovisioning of more than 300 % of the video bit-rate. In any other case, quality switching always occurs.

In the second part of the study we discuss the efficiency of the adaption strategy from a networking view point. The results show that by replacing lower quality segments, the YouTube player is able to increase significantly the average playback quality by up to 0.7 quality levels. However, 30 % redundant data has to be downloaded for this. Based on the amount of redundant traffic, we estimate the optimal, i.e. highest, average quality level which could be achieved by downloading the same total amount of data but avoiding redundant traffic. The results show that without redundancy the amount of downloaded data could have been used to increase the average quality level by up to 1.3 quality levels. Therefor, about a half quality level is unnecessarily downloaded and discarded due to the redundancy.

In the future, we plan to include highly varying bandwidth conditions, e.g. as found in cellular access. Furthermore, the maximum achievable playback quality for a certain data volume can be formulated as an optimization problem instead of regression based on historic data. This can give insight into an upper bound for potential improvements to YouTube's adaptation algorithm.

References

[1] "Cisco visual networking index: Foreecast 2014 - 2019." [Online]. Available: http://goo.gl/0iqjo9

[2] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *Communications Surveys Tutorials, IEEE*, 2014.

[3] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet video delivery in youtube: from traffic measurements to quality of experience," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 264–301.

[4] T. Hoßfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia, "Identifying qoe optimal adaptation of http adaptive streaming based on subjective studies," *Computer Networks*, vol. 81, pp. 320–332, 2015.

[5] Sandvine, "Global Internet Phenomen Report 1H 2014," 2014.

[6] C. Sieber, A. Blenk, M. Hinteregger, and W. Kellerer, "The cost of aggressive HTTP adaptive streaming: Quantifying YouTube's redundant traffic," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, may 2015, pp. 1261–1267.

[7] C. Sieber, T. Hoßfeld, T. Zinner, P. Tran-Gia, and C. Timmerer, "Implementation and User-centric Comparison of a Novel Adaptation Logic for DASH with SVC," in *IFIP/IEEE International Workshop on Quality of Experience Centric Management (QCMan)*, Ghent, Belgium, May 2013.

[8] J. Añorga, S. Arrizabalaga, B. Sedano, M. Alonso-arce, and J. Mendizabal, "YouTube's DASH implementation analysis," in *19th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, 2015, pp. 61–66.

[9] L. Yao, W. Qi, G. Lei, S. Bo, C. Songqing, and L. Yingjie, "Investigating Redundant Internet Video Streaming Traffic on iOS Devices: Causes and Solutions," *Multimedia, IEEE Transactions on*, vol. 16, no. 2, 2014.

[10] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A comparative study of android and ios for accessing internet streaming services," in *Passive and Active Measurement*. Springer, 2013, pp. 104–114.

[11] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth, "Characterizing Client Behavior of Commercial Mobile Video Streaming Services," in *Proceedings of Workshop on Mobile Video Delivery, MoViD'14*, 2014.

[12] H. Nam, B. H. Kim, D. Calin, and H. G. Schulzrinne, "Mobile video is inefficient: A traffic analysis," 2013.

[13] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 2011.

[14] M. Ito, R. Antonello, D. Sadok, and S. Fernandes, "Network level characterization of adaptive streaming over http applications," in *IEEE Symposium on Computers and Communication (ISCC)*, June 2014.

[15] S. Alcock and R. Nelson, "Application flow control in youtube video streams," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, Apr. 2011.

[16] "mitmproxy." [Online]. Available: https://mitmproxy.org/

[17] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "YoMo: A YouTube Application Comfort Monitoring Tool," in *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications*, 2010.

[18] "youtube-dl." [Online]. Available: https://rg3.github.io/youtube-dl/

[19] "ffprobe." [Online]. Available: https://ffmpeg.org/ffprobe.html

[20] "Traces and programs used in this paper." [Online]. Available: http://git.io/vRSSW/

[21] R. E. Barlow, D. J. Bartholomew, J. Bremner, and H. D. Brunk, *Statistical inference under order restrictions: the theory and application of isotonic regression*. Wiley New York, 1972.

Clique-Aware Mobile Social Clouds

Christian Quadri, Matteo Zignani, Sabrina Gaito, Gian Paolo Rossi
Computer Science Dept. , University of Milan
Email: firstname.lastname@unimi.it

***Abstract*—The important role played by cliques in identifying cohesive subgroups of people has been theorized and explored by sociologists years ago, but only recently investigated in large-scale social networks. In this paper we focus on the interplay between cliques established by on-phone communications and the urban locations their members share each other. The results about co-located cliques has been achieved through the extensive analysis of a large anonymized dataset of Call Detail Records (CDR) relying on the phone activities of nearly 1 million people in the city of Milan. Taking the cue from the observation of cliques, the paper envisions and designs a novel clique-support service for mobile users by fully exploiting the current virtualization process that is radically transforming the core network of mobile operators. The approach we propose brings together a few important contributions: first, it concretely shows that the current NFV-enabled trend of placing cloud services at the edge of the operator's network is viable and may have a payoff in terms of traffic offloading and improved user's experience; secondly, it demonstrates for the first time that a few typical cloud-based services can effectively be directly performed inside the mobile network by simply leveraging the rich amount of data about users' location and mobility behavior.**

I. Introduction

The real-life sociality of each individual unrolls among friends with strong social ties, is still bound to proximity, which means being near enough to repeatedly encounter the person and to do things together, and gives opportunities to share common ideas and personal feelings with each other. Today's social network operators and service providers willing to deploy cloud-based platforms suitable for supporting real life social interactions are required to tighten their grip on the daily life of their users and keep some synchrony with their social and mobility behaviour. Such a direct drive with the daily life of individuals can only be achieved by performing continuous monitoring of the users' devices and becoming aware of their surroundings through the connectivity infrastructure provided by mobile operators. The result is a growing amount of resource draining interactions that reduce the users' quality of experience, force mobile operator to continuous expensive upgrades and give rise to critical privacy concerns.

The next generation cellular networks, with flexible and decentralized architectures more akin to modern data centers, will provide the connectivity and computing infrastructure where all system and networking issues can be properly addressed, mitigated and solved [1], [2], [3]. However, to benefit of this emerging new service provisioning in the design of services inspired by and tailored on individual's behavior, the research is today challenged by the urgency of better understanding how real life sociality is undertaken and properly drive the design of a digital service and of the underlying network infrastructure, accordingly. In line with these arguments, this paper pursues the following objectives:

- give empirical evidence of the nature of the social interactions among people in real life. We achieve this goal through an extensive analysis of a large anonymized dataset of Call Detail Records (CDR) relying on phone activities (voice, data and text) of nearly 1 million people in the city of Milan.
- define functional and system requirements of a digital platform supporting real life social activities, interactions and encounters.
- envision a service architecture suitable for addressing these requirements. In line with a mobile edge computing approach, we sketch a viable virtualized approach placing the service as close as possible to the end-user at the edge of the mobile operator's core network.

The main contributions of the paper are:

i. We observe that the human sociality mediated by on-phone communications is organized in cliques of small size (we observed cliques ranging from 3 to 9 people) and with intense internal interactions revealing strong social ties among people belonging to the clique. People in a clique are used to encounter periodically in a variety of city's locations, thus proving the importance of physical encounters in real life sociality and the role of on-phone interactions on capturing mobi-social groups. The interplay between on-phone groups and group meeting is stronger w.r.t. results about a similar correlation measured on single links [4].

ii. Cliques justify the rise of CLique-Aware Mobile Social networks, we name it CLAMS, serving the small community of individuals and providing internal basic services, such as clique interactions, sharing of contents relevant to the clique (namely, photos, video or other contents), and clique specific privacy preservation policies driven by explicit user's consent. During extemporaneous encounters of the persons in a clique, devices' proximity may be opportunistically exploited to provide CLAMS communications. This way CLAMS creates a targeted service which supports the needs of the cliques;

iii. The potentially huge amount of CLAMSs (more than 53.000 have been observed in our dataset) has explicit entailments on system requirements when supporting the interactions between user's device and cloud-based CLAMS service. Efficiency, flexibility and scalability requirements

ISBN 978-3-901882-83-8

ISBN 978-3-901882-84-5, IFIP Networking 2016

advocate a virtualized approach enabling to dynamically place contents as close as possible to the mobile users, and to lower the entire cloud-based CLAMS service next to the users when they encounter each other in a city location. In the perspective of a totally virtualized mobile network, the cloud service may not only be placed next to the operator's data centers, but it can definitely enter it and be placed at dynamically changing levels in the network hierarchy with the aim to optimize resource consumption, reduce latency and ensure traffic offloading from the operator's core network. In line with these arguments, we let the cloud service cooperate with the operator's core functionalities to orchestrate the placement of the CLAMS cloud thus finding the best trade-off between network resource consumption and user's perceived service quality. This way, for instance, when the members of a clique are co-located, the cloud service can be placed at the edge of the core network [5], [2] to ensure low latency interactions and efficient content sharing. Co-location is frequent among friends and is common in workplaces, where members of a team daily share the same area and perform intense interactions. When the devices of a clique happen to be under the coverage area of a single cell tower, the relevant device-to-device communications can be seamlessly performed via Direct-LTE [6] or opportunistically. The paper describes a NFV/SDN-based architecture supporting dynamic placement of CLAMS through cooperation between network and cloud operators.

II. Mobi-social groups

A cloud-based platform supporting social interactions cannot disregard the role of social groups as the main constituent of its infrastructure. Social groups are often identified by the notion of cohesive subgroups, i.e. subsets of individuals among whom there are frequent and relatively strong interactions. In these groups beliefs, interests and idea are often very homogeneous due to the pressure towards uniformity and group standards exerted by intense interactions [7]. Favorite places are among the interests of a cohesive group. In fact, shared places encourage the formation and the strengthening of social relationships and, conversely, groups could choose a specific place to better express themselves.

Through mobile phone data we are able to highlight the interplay between cohesive groups and people mobility. Although on-phone communications capture a part of all social interactions, mobile phone data are inclined to trusted communications since people are not willing to share their private phone number with everyone. This way we rely on calls and text-messages to identify cohesive groups. At the same time we exploit the localization provided by the network infrastructure to reconstruct the mobility patterns of the group members. Specifically we proceed in our analysis by *i*) identifying close groups through the extraction of maximal cliques; *ii*) studying how interactions inside a group are distributed and finally; *iii*) mapping a clique to the places it visits, finding what we call 'mobi-social' groups.

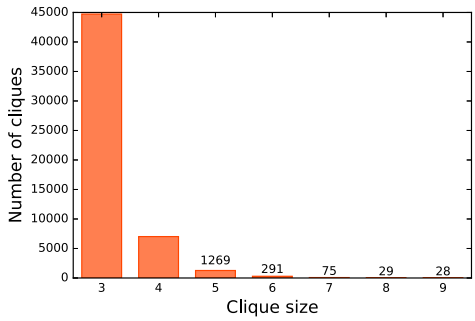


Fig. 1: Clique size distribution.

A. Dataset

Our mobile phone dataset [8] consists of Call Detail Records (CDRs) containing voice-call, text and data activities of nearly 1 million mobile subscribers in the Milan metropolitan area. The records span over 67 days, from March 26 to May 31, 2012; a period long enough to reconstruct most of the on-phone social relationships, as observed in Onnela *et al.* [9] (the statistical characteristics of the network largely saturate in a two-months-long sample). Whenever a voice/text call is issued, a CDR entry is created as a 6-ple $t_{CDR} = \langle s, r, t_{start}, d, cell, area \rangle$, where s and r respectively represent the sender and the receiver of the call/sms, t_{start} is the initial time of the activity (when the call starts or a SMS is sent), d is the duration and $cell$ is the serving cell the user s is attached to. The field $area$ indicates the location-name attribute related to the $cell$, e.g. street/square name or city's zone, that represents a coarse grain division of the city region.

We discovered that nearly 40% of calls have duration equal to 0. Besides missed or unanswered calls, 0-duration calls are reckoning with a common practice in Italy to use rings for implicit communications ("Call me back soon", "I'm just arrived", etc.). Due to the ambiguity of 0-duration calls in the analysis of social interactions, we discarded them from the dataset which finally turns out to be composed of 41 millions calls and 20 millions SMS. We also filtered out calls involving other mobile operators, both incoming and outgoing, thus maintaining only activities involving subscribers of the same operator. This way we eliminate the bias between operators; in fact, we have a full access to the call/SMS records of one operator, while partial access to the calls issued towards subscribers of other competitors.

Unlike previous studies where cell tower may cover a zone as wide as a few kilometers [10], the dataset we are leveraging reports data about cell towers inside a city space where a very small coverage radius, of one or few hundred meters, is adopted. This characteristic, combined with the knowledge of cliques, is a powerful enabler to study the off-line social life of tight-knit groups as information on both their communications and meetings can be mined from the dataset. In fact, we argue that, when people with strong social relationships happen to be contemporaneously co-located, they are more likely to have a social encounter and a face-to-face interaction than being co-

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

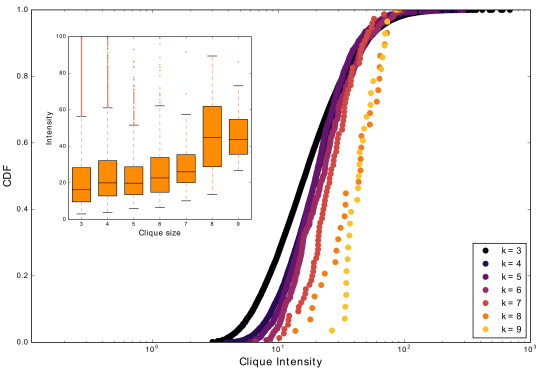


Fig. 2: Distribution (CDF) of the clique intensity as a function of the clique size (k). The inset figure shows the correlation between the clique size and the median clique intensity. The boxes contain values between the first and the third quartile.

located by chance.

B. Clique analysis

We construct the call/SMS undirected weighted graph $G(N, E, W)$ where N is the set of the mobile operator subscribers and E is the set of ties between them, where a link exists between node i and node j if at least three communications, either voice call or text messages, with an overall duration greater than a minute, have been exchanged between the two end-nodes. The order of the resulting network is 289448, while its size is 429273. The weight $w_{i,j}$ of a link (i, j) is an integer number equal to the overall number of communications, voice calls and text messages, between the two nodes.

We rely on the notion of clique to identify cohesive subgroups on the mobile phone graph. To this aim on $G(N, E)$ we performed an adaptation of the Bron and Kerbosch's algorithm [11] to find all the maximal cliques in an undirected graph, i.e. the maximal fully connected subgraphs in the network. Although the worst-case running time of the algorithm is exponential on the number of nodes, in practice it has been demonstrated to run very fast on real networks. The algorithm returns a set C of maximal cliques, whose elements will be indicated by c_i.

1) Do cliques exist?: Overall, we observed 53437 maximal cliques whose size (k) is greater than 2. 122.027 people, 29.4% of all users, are involved in at least one clique, a very high percentage if we mind the limits of mobile phone data: only communications between users of the same mobile operator and on a limited geographical area are recorded. The same holds for links since the 44% of network ties are intra-clique. This very first result confirms the presence of strongly cohesive groups in mobile phone graphs and highlights the importance of cliques in the people's sociality expressed through on-phone communications.

The distribution of clique size is in line with other studies [9], [10], [12], too. As shown in Figure 1, very small cliques ($k = 3, 4$) are predominant. They represent the reference group from where an individual is particularly likely to seek advice or support when needed since the corresponding links represent very strong social relationships. In particular the prevalence of triangles ($k = 3$) highlights that, also in call graphs, the triadic closure process is very likely to happen and act on the network structure. Fig.1 also indicates that larger cliques are less widespread and they more likely represent cohesive groups of interest than familiar or friendship tight groups. No cliques larger than 9 was observed, however we suppose that larger cliques could actually be observed by combining phone data of other mobile operators.

2) Clique strength: One of the properties characterizing cohesive subgroups is the higher frequency of ties among their members compared to the remainder of the network. In line with the previous works on cliques in call graph [9], we compute the clique intensity int to assess the propensity of communicating inside a clique. The clique intensity of a clique c_i is defined as the geometric mean of its link weights:

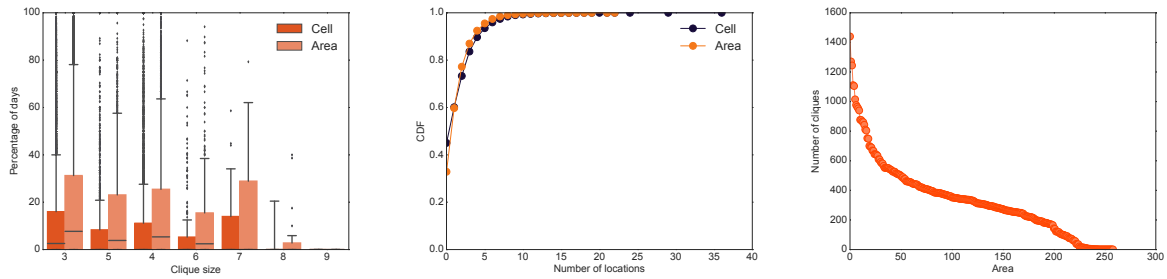
$$int(c_i) = \left(\prod_{(i,j) \in E(c_i)} w_{i,j} \right)^{1/|E(c_i)|} \quad (1)$$

where $E(c_i)$ denotes the links forming the clique c_i. Fig.2 shows the cumulative distribution function (CDF) of the clique intensity for the different values of the clique size. Globally, we observe that members in the cliques are quite interactive since the median is equal or greater than 20 interactions for each distribution. Thus, on average, each pair in a clique has communicated more than 20 times in two months. Moreover the range of the intensity reduces as the clique size increases. For instance, triangles include both scarcely (3 interactions per pair on average) and very active (> 600 interactions per pair) 3-ples, while the intensity of larger cliques lies within a smaller range ($25 - 60$ interactions per pair).

As shown in the inset boxplot of Fig.2, we make the correlation between the clique size and the intensity more explicit. The median intensity increases as the size increases, specifically it doubles for the largest cliques. From the network operator viewpoint these large cliques represent a strategic target (marketing, premium services) due to their frequent communications and their cohesiveness, while from a sociological viewpoint the increasing trend of the intensity suggests that maintaining large strongly cohesive groups can be highly demanding in terms of communications and resource consumption, thus advocating specific support by the cellular network infrastructure.

3) Clique meeting places: Mobile call data have been exploited in recent years to extract the relation between people's sociality and mobility. Most of researches focused on dyads and found out that people having a social relationship are more likely to share spaces than unrelated ones, although all these studies have been using a coarse spatial granularity in the mile/km scale. In [13] we observed a weak correlation between social and spatial communities as detected on the same dataset we are using here with a finer granularity ensured by the dense

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



(a) (b) (c)

Fig. 3: a) Percentage of cliques that meet. b) Distribution of the number of locations (cell/area) per clique. c) Number of cliques meeting in each area.

placement of urban cell towers. Nonetheless, communities remain quite loose organizations that only sporadically gather in a place to perform common activities. By contrast, at the network mesoscopic scale, cliques are more interesting when a strong socio-spatial connotation is needed. We adopt the following methodology to detect clique co-location. For each clique we reconstruct the mobility trace of each user starting from the CDRs, and we transform it as follows: we convert each point, identified by triplet $\langle t, cell, area \rangle$, in a time interval assuming that if the user was in a cell at time t she/he was in that cell from $t - \Delta$ until $t + \Delta$. In this paper we use Δ = 30 minutes in line with [4]. Each time interval maintain the same location attributes (cell and areas) of the trace point from which is generated. Then we merge all the traces of the members of the clique, by retaining only the time overlapping intervals that share the same location attributes. In the following we consider two levels of clique co-location, cell-tower and area. Cell co-location, i.e. the strictest one, implies that, for all the retained time interval, the cell attribute must be the same across all clique members, while in case of area co-location this restriction is limited to the area attribute.

Our findings show that 57.1% of cliques meet at least once in the considered time frame, that is an impressively high amount when considering the network sparsity. The result confirms that there is a strong correlation between on-phone interactions and real life encounters. The percentage of meeting cliques per clique size is reported in Fig.3a. The likelihood of having all clique members gathered in a place at the same time decreases with larger cliques.

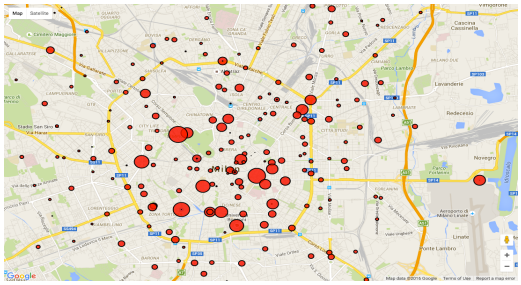


Fig. 4: City map with places of encounter.

4) Social urban spaces: In Fig.3b and Fig.3c the number of locations visited by a clique and the number of cliques meeting in each area are reported, respectively. A visual representation can be seen in Fig.4 where most social urban spaces are reported on the city map. Fig.3c and the map in Fig.4 highlight the fact that some city locations are more favored that others for clique encounters and therefore, indirectly, that cities have places more social than others.

III. NFV-BASED ARCHITECTURE FOR CLAMS

The results of the CDR analysis described in the previous section give a quantitative evidence of how social interactions are performed in real life. We show that interactions are often performed within cliques of individuals with strong social/professional ties and that persons engaged in a clique are also used to encounter each other in different locations of a city, i.e. they are co-located. Interactions in a clique are performed either by voice call or text message, or generate data traffic when contents are shared. Traffic is generated when the persons are spread in different locations of the city and continues even when they are co-located.

In such a scenario, we can envision that the traffic load created by the interactions within a clique is intended to grow sharply with the growth of people sharing resource-draining contents (for both leisure and work), playing distributed games, or as soon as virtual reality will become a common tool for social engagement. Mobile operators are already designing the next generation cellular networks to achieve higher standards of flexibility, scalability and performance through virtualized architectures more akin to data centers. This radical transformation will provide the connectivity and computing infrastructure where all these emerging system and networking issues can be solved. Similarly, cloud-based operators will be urged to collaborate with network operators to ensure the best quality of experience to their users, while optimizing resource consumption and reducing time latencies among interacting people. As shown by this paper, cliques are social aggregations deserving this type of care by any cloud-based service provider.

In this Section we provide the description of the architecture exploiting the underway evolution of cellular networks to properly support interactions among people belonging to a clique, i.e. we provide the CLAMS supporting architecture.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

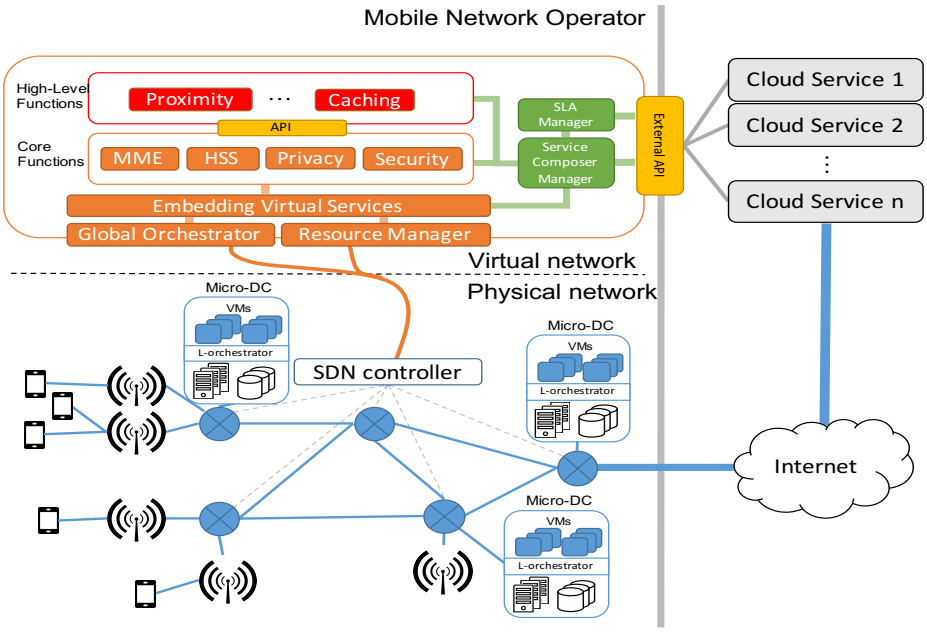


Fig. 5: CLAMS architecture.

At the hearth of this idea lays the need to provide a framework enabling network and cloud operators to negotiate the service agreement, package it with the instantiation of the cloud service and let the network operator to orchestrate the placement of the embedded service in order to satisfy the service requirements and optimize the resource utilization. The architecture is coherent with the current evolution of mobile networks to meet the 5G requirements of flexibility, scalability, small delay and high bandwidth. Our approach has been inspired by [14] where a new architecture, based on network function virtualization (NFV) and software defined networking (SDN), is presented without requiring control and data planes; rather, they are tailored according to the requirements of specific applications and devices. In line with the proposal, any network function can be dynamically instantiated within the cloud infrastructure to satisfy specific application and device performance and relevant functional requirements. Relying on this approach, our proposal enables CLAMS to be easily deployed and managed in a virtualized environment.

A. Architecture overview

The Fig. 5 shows an overview of the architecture supporting the CLAMS service. The network architecture combines both *Physical* and *Virtual* elements. The physical network contains all resources – computational, storage and networking – providing connectivity to the users, and implements the data plane protocols. The network nodes are compatible with SDN standards, e.g. Openflow, and are all managed by a SDN controller. Inside the physical network we place micro data centers (Micro-DC), each handling one or more access nodes and covering different geographical areas (preferably overlapped to favor load balancing). The network operator could decide to assign a Micro-DC to a single area, or Tracking Area, or even to a single cell tower, for dealing with the traffic burden and requirements that crowded locations of a city bring together, like the case of the most social places we identified in the previous section or whenever popular events are organized in a city. Micro-DCs have computational and storage resources managed by a local orchestrator (L-orchestrator) and they are able to host virtual instances of network functions (e.g. MME, HSS and security) as well as virtual packages embedding cloud services and serving cliques of users, either spread in different locations of the city or co-located.

On top of the physical network, the virtual network is composed of different modules in charge of managing the physical network and orchestrating the virtual services that, in the perspective of this proposal, can be both internal and external to mobile operator. The three modules at the edge between physical and virtual networks have the responsibility of managing physical resources. In particular, the *Global Orchestrator* supervises the allocation and the deployment of virtual resources, the *Resource Manager* collects and processes all status information of the allocated resources, the *Embedding Virtual Services* maps the virtual services on the physical resources on the base of the information provided by the resource manager. A set of *Core Functions* lays on top of the edge modules and constitutes the template of the current LTE core network functionalities. These templates will be configured and combined in service packages according to the global orchestration strategy during the deploying phase. Core Functions may be combined with *High-level Functions* to provide templates of added-value services, whose configuration is still performed by the global orchestrator during the deploying phase. Moreover, cloud service providers can seek improved user's quality of experience and service provisioning by requiring to wrap cloud-specific functions in combination

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

with both core and higher functions. The obtained service package is thus embedded and deployed by means of the edge services. For instance, a clique of CLAMS can be instantiated in a package together with Proximity and Caching Higher Functions, and with most of the Core Functions.

The cloud-based service provider can interact with the mobile operator through the *Service Composer Manager* that offers a sort of dashboard interface from where the cloud provider is able to wrap and tailor services by simply leveraging functions and services provided by the network operator. The Service Level Agreement module, *SLA manager*, has been added to regulate the usage of the infrastructure. Both Service Composer Manager and SLA Manager are accessible through a set of API and are needed to negotiate the service quality and compose the cloud service to deploy inside the operator's network.

B. CLAMS services deployment operations

When the CLAMS service provider needs to wrap a clique-specific package inside the operator's network, it directly accesses the service composer manager. The service-specific requirements drive the selection of different cloud and network functions and lead to create the template of the CLAMS service that, if it satisfies the policies defined in the SLA Manager, can be added to the repository – one per each cloud service – of the deployable CLAMS services. While packaging the cloud service, the requirements in terms of bandwidth, delay, computational and storage capabilities are specified.

The cloud service provider can also specify a set of rules, within the negotiated Service Level Agreement, to apply when the service is deployed, e.g. to optimize content access and interactions when clique members are co-located, or to characterize the traffic among clique components. The activation of a set of rules triggers the deployment phase, a message is sent to the Embedding Virtual Service module that performs the embedding of the required CLAMS service into the physical network. The Global Orchestrator is then activated, operates the deployment of the virtual instances and informs the SDN controller about the required links. At this point, the CLAMS service is ready to be used by users.

Although the described architecture enables to sink cloud services into the depths of the operator's core network, no sensible information are, interestingly, shared between the two operators; for example, the users' location remains in the domain of mobile operators, while the cloud service provider is unaware of where the CLAMS and relevant users are placed. Moreover, the rules to preserve privacy are managed by the cloud operator and deployed accordingly.

IV. Concluding remarks

In this work we show that on-phone interactions bring together the formation of cohesive social groups of persons, or cliques, and we give evidence of them. Moreover, we show that people engaged in cliques are also very likely to meet each other in different locations of the city space, thus enabling the identification of the most social locations of the city, i.e. the places where cliques are more likely to encounter. These information are relevant whenever new socio-techno-driven services have to be designed and deployed by either mobile or cloud operators. By exploiting the underway radical transformation of the core network of mobile operators, we provide a virtualized network architecture enabling cooperation between mobile and cloud operators to embed cloud services into the mobile network thus ensuring high user's experience, reducing resource consumption and minimizing communication latency. All these first results are promising enough to induce us to perform further investigations. In particular, a deep temporal analysis of clique's socio-spatial patterns is needed to better design the architecture, while specific simulation should be performed to quantify the benefits achieved by adding flexible placement in Micro-DC by means of function virtualization.

References

[1] Ericsson, "The real-time cloud," White Paper, February 2014.

[2] A. Manzalini, R. Saracco, C. Buyukkoc, P. Chemouil, S. Kukli?ski, A. Gladisch, M. Fukui, E. Dekel, D. Soldani, M. Ulema, W. Cerroni, F. Callegati, G. Schembra, V. Riccobene, C. Mas Machuca, A. Galis, and J. Mueller, "Software-defined networks for future networks and services," in *White Paper based on the IEEE Workshop SDN4FNS*. IEEE, 2013.

[3] H. Wang, S. Chen, H. Xu, M. Ai, and Y. Shi, "Softnet: A software defined decentralized mobile network architecture toward 5g," *Network, IEEE*, vol. 29, no. 2, pp. 16–22, March 2015.

[4] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi, "Human mobility, social ties, and link prediction," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: ACM, 2011, pp. 1100–1108. [Online]. Available: http://doi.acm.org/10.1145/2020408.2020581

[5] European Telecommunication Standards Institute (ETSI), "Mobile-edge computing - introductory technical white paper," Technical Report, September 2014.

[6] 3GPP TR 36.877, "LTE Device to Device Proximity Services; User Equipment (UE) radio transmission and reception (Release 12)," Technical Report, 2014.

[7] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge university press, 1994, vol. 8.

[8] C. Quadri, M. Zignani, L. Capra, S. Gaito, and G. P. Rossi, "Multidimensional human dynamics in mobile phone communications," *PLoS ONE*, vol. 9, no. 7, pp. 1–12, 07 2014.

[9] J.-P. Onnela, J. Saramki, J. Hyvnen, G. Szab, D. Lazer, K. Kaski, J. Kertsz, and A.-L. Barabsi, "Structure and tie strengths in mobile communication networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 18, pp. 7332–7336, 2007. [Online]. Available: http://www.pnas.org/content/104/18/7332.abstract

[10] A. A. Nanavati, R. Singh, D. Chakraborty, K. Dasgupta, S. Mukherjea, G. Das, S. Gurumurthy, and A. Joshi, "Analyzing the structure and evolution of massive telecom graphs," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 5, pp. 703–718, 2008.

[11] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, no. 1, pp. 28 – 42, 2006.

[12] M.-X. Li, W.-J. Xie, Z.-Q. Jiang, and W.-X. Zhou, "Communication cliques in mobile phone calling networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2015, no. 11, p. P11007, 2015.

[13] M. Zignani, C. Quadri, S. Gaito, and G. P. Rossi, "Calling, texting, and moving: multidimensional interactions of mobile phone users," *Computational Social Networks*, vol. 2, no. 1, pp. 1–24, 2015.

[14] R. Trivisonno, R. Guerzoni, I. Vaishnavi, and D. Soldani, "Sdn-based 5g mobile networks: architecture, functions, procedures and backward compatibility," *Transactions on Emerging Telecommunications Technologies*, vol. 26, no. 1, pp. 82–92, 2015. [Online]. Available: http://dx.doi.org/10.1002/ett.2915

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

On the Technical and Social Structure of Community Networks

Leonardo Maccari
Department of Information Engineering and Computer Science,
University of Trento, Italy
{leonardo.maccari}@disi.unitn.it

***Abstract*—A Community Network is a bottom-up network created by a community of people with the goal of building a self-owned, self-managed communication infrastructure. Community Networks are blooming, they range from small ones (tens of nodes) to gigantic ones (tens of thousands of nodes), they are made primarily of wireless links but in some cases they mix wired and wireless technologies. People running a Community Network wish to have more independence and more control on the infrastructure compared to what commercial ISPs offer. Such networks can not be understood without studying the interaction between the social and the technical aspects, since both layers are tightly intertwined. This paper will show some properties of three community networks, indicating that they have some structural peculiarities. It will also show a socio-technical analysis using data coming from the public mailing list of one of the community, in order to highlight issues that the community needs to address to guarantee its sustainable growth.**

I. Introduction

A Community Network (CN) is a communication network set-up by a community of people with a bottom-up, participatory approach. It is primarily a wireless mesh network, extended in some cases with wired connections. While the concept of CN is not new [1] their development in the last period was remarkable, and today, CN represent an extremely interesting and timely research topic due to (among others) three factors[1]: The first is that recent standards (such as IEEE 802.11n/ac) make it possible to realize high-capacity wireless links that can reach a length up to tens of km. This allows to create networks that cover entire cities with excellent performances. Thus, the scalability and the protocols for mesh networks that were of high interest in the 2000s, today can be finally applied to real networks made of hundreds of nodes.

The second factor is that CNs have shown to be an effective way to bring connectivity in underserved areas, so they are a valuable instrument against digital divide. An outstanding case is the Guifi.net CN that was awarded by the European Commission with the 2015 European Broadband Awards[2].

The third factor is that CNs are unique environments to experiment inter-disciplinary research, since they can be better interpreted combining different research methodologies. In a CN each network node corresponds to a person, a family, an association or a small business. The management of the network is collective so the social dynamics inside the community influence the technical choices about the network itself. It is crucial to understand the motivations that drive the communities and the social norms that regulate them if one wants to propose solutions that are not only technically sound but also compatible with the open and participatory nature of a CN. Such a mix of social and technical aspects makes a CN an example of the emerging paradigm of the so-called "Internet of People" (IoP), because the whole network infrastructure is shaped by the behaviours of the individuals and their group decisions. It is thus very interesting to understand if this original organization leads to networks with different features compared to other networks that have been already studied.

This paper contributes to this discussion and will analyse the data available from three community networks: the FunkFeuer network in Wien and Graz, and the ninux.org network in Rome (abbreviated respectively as FFWien, FFGraz, ninux). The goal of the paper is to answer two questions:

- Is the evolution of the network graph different compared to other communication networks, such as scale-free networks?
- Given that the goal of the community is to build a distributed network with a de-centralized management, is the result close to the expectations of the community?

II. Motivations, data-set, background

There is a large body of works that suggest that many network graphs, including the Internet, show a scale-free behaviour, that is, the distribution of the degree of the nodes follows a power-law function. A scale-free network presents a small number of densely connected hubs that strongly influence the behaviour of a communication network. Hubs guarantee that the average shortest path grows slowly with the number of nodes, which is a positive factor because it keeps the round-trip-time low. However, hubs are enormously more important than other nodes, which is a negative factor, because

This work was financed partially by the University of Trento under the grant "Wireless Community Net-works: A Novel Techno-Legal Approach" —Strategic Projects 2014, and partially by the European Commission, H2020-ICT-2015 Programme, Grant Number 688768 'netCommons' (Network Infrastructure as Commons).

[1]Recently, various large research projects focussed on CNs: see the CONFINE, CLOMMUNITY, P2PValue and netCommons projects, respectively at www.confine-project.eu, www.clommunity-project.eu, www.p2pvalue.eu and www.netcommons.eu

ISBN 978-3-901882-83-8

[2]see https://ec.europa.eu/digital-agenda/en/news/five-projects-got-first-ever-european-broadband-award

ISBN 978-3-901882-84-5, IFIP Networking 2016

they introduce points of failures. This paper uses data from the FFWien and FFGraz network to verify if this pattern can be observed also in CNs.

Another level of interpretation is given by the socio-technical analysis of ninux, that gives insights on the sustainability of the network itself. Recent social analysis [2] have shown that community networks have strong political motivations: the construction of an independent, robust, decentralized network infrastructure. Ninux is not an exception [3]: Ninux participants have a critical opinion of ISPs and service providers motivated by the recent discussions about neutrality, privacy and forced disconnections. They identify the root cause of these problems in the centralization (both technical and administrative) of the networks and of the services, and for this reason they build their own decentralized network, managed with a peer-to-peer approach. Peer-to-peer organization is a key feature of ninux: since the mesh network works without introducing hierarchies and layers, the community tries to reflect this approach also in the social organization. Thus, the ninux community did not create a formal association, it does not assign formal responsibilities and does not have "roles" assigned to people. The discussions in the community are primarily carried on in the mailing lists and in weekly face-to-face meetings, and decisions are taken with a consensus-based method. This approach is shared with other CNs and it is original in the communication panorama.

It is legitimate to ask what is the degree of success of ninux and its overall sustainability. To answer, it is crucial to remember that the network exists because the participants, through their social interactions, cooperate to reach a common goal so the social networking layer is as important as the technological one. If the network is not technically sound, it will fail in bringing services to the people, but also if the community is not participated enough, there will be a lack of the social capital needed to maintain the infrastructure. The key observation is that in a CN the social network and the network infrastructure can be linked, since every node belongs to a person. The two layers of analysis can be explored with the same instruments to understand if there are cross-layer single points of failure that can mine the future growth of the network.

A. The data-set

The three networks use OLSR (Optimized Link-State Routing), a link-state routing protocol that makes it possible for each node to be aware of the whole network topology. The communities publish the network topology dumped by the OLSRd daemon, that can be used to analyse the network evolution. The topology recorded by the routing daemon can be misleading: in some cases a number of devices placed in the same physical location are attached to a wired switch and each of them runs a separate instance of the routing protocol. For OLSR they are different nodes but, in practice, they are not. To merge these cliques, another source of information is needed. More details about the networks and the merging technique are out of the scope of this paper, the interested reader can find details in the published source code[3] and in previous works [4][5].

	FFWien	FFGraz	ninux
maximum recorded nodes	235	126	140
maximum recorded links	450	181	158
time series available	yes	yes	limited
first dump	2013-07-27	2007-03-31	2014-1-14
last dump	2014-02-15	2016-02-21	2014-1-20
dump interval	weekly	monthly	every 5 min
node ownership	no	no	yes
mailing list	no	no	yes

TABLE I: The summary of the available data

The FunkFeuer networks publish a long history of dumps, while for Ninux only the current state is available, plus data collected in a week-long monitoring realized in 2014 [5]. Tab. I reports a summary of the data used for this paper. In the rest of the paper the time based evolution of the network always refers to the FreiFunk networks, instead when the analysis is done on a single snapshot, the sample with the largest number of nodes for each network in considered.

For the ninux network, two other sources of information were accessed. The first is a database containing the mapping between the physical node and the ID of a person that owns it, the second is the archive of the mailing lists of the ninux community of Rome for the year 2014.

B. Related Works

CNs have been the subject of a series of research works in the past years that had the goal of analysing their topological features [6][4][5][7] their routing solutions [8][9] and their social and management aspects [10][11]. This paper performs a different analysis based on two original elements, the first is the analysis of the time-evolution of the networks, which helps understanding what was, and potentially what will be the evolution of the network. The second is the mixed social and technological analysis aimed at identifying single points of failure in the techno-social organization of the network.

III. The Network Graphs and their evolution

Fig. 1 Fig. 2 and Fig. 3 show the relative frequency of the degree distribution for the three networks, and the best-fit with a power-law function. A power-law degree distribution is normally observable in the central part of a distribution or in the right tail. In this case the size of the networks (hundreds of nodes, and maximum degree that ranges from 11 to 29) makes it statistically hard to identify a trend.

An alternative approach is to investigate if the network evolution follows a preferential attachment model, which would lead to a more evident scale-free behaviour with the growth of the network. The preferential attachment model describes the way in which new nodes are added to the network, and the entry points they connect to. In such model the rate $\Pi(k)$ with which a node with k links acquires new links is a monotonically increasing function of k. Following a preferential attachment model is not a necessary condition to

[3]accessible via Git at github.com:leonardomaccari/

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

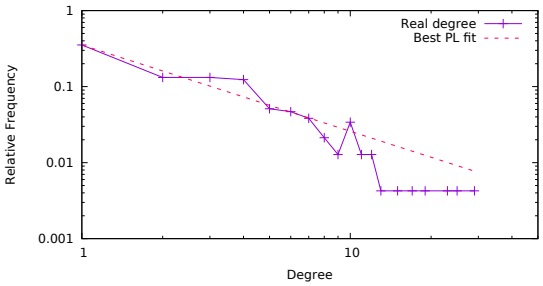


Fig. 1: The degree distribution for FFWien, and the best power-law fit $x^{-\alpha}$, $\alpha = 1.13$.

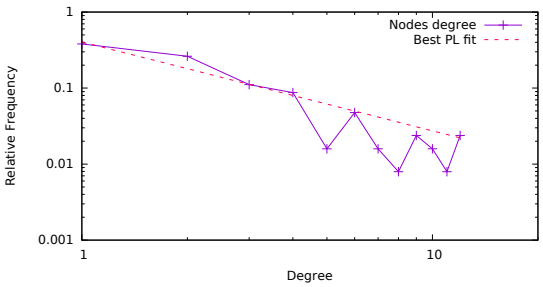


Fig. 2: The degree distribution for FFGraz, and the best power-law fit $x^{-\alpha}$, $\alpha = 1.16$.

have a scale-free network, however since it has been shown that it is at the base of several different kinds of scale-free networks (the Internet graph for instance, has been shown to have $\Pi(k) \propto k$ so that the probability of acquiring now links is proportional to the current number of links of a node [12]) measuring the relationship between $\Pi(k)$ and k can give insights on the future evolution of the network. This behaviour is easier to test on this data-set because the total number of new nodes that joined the network during the observed period is much higher than the number of nodes at the end of the interval, since many nodes join the network for a

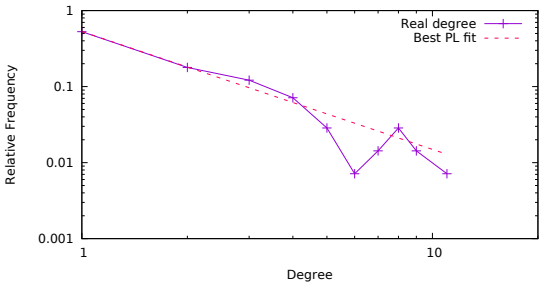


Fig. 3: The degree distribution for ninux, and the best power-law fit $x^{-\alpha}$, $\alpha = 1.55$.

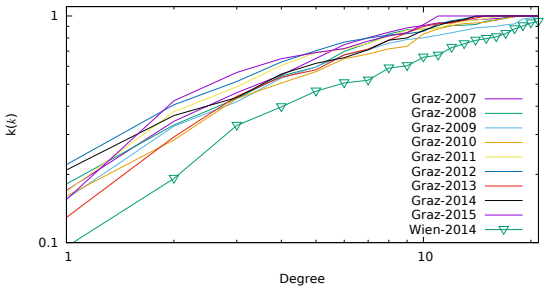


Fig. 4: The value of $\mathrm{k}(k)$ in the FFGraz and FFWien networks, separately computed per each year

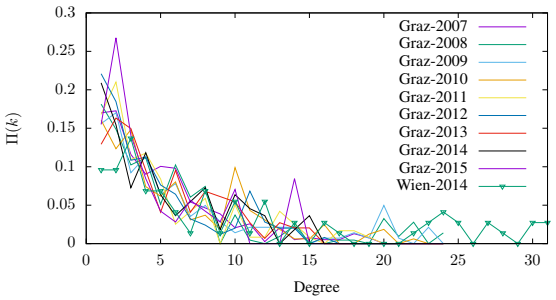


Fig. 5: The value of $\Pi(k)$ in the FFGraz and FFWien networks, separately computed per each year

limited period of time. To test the hypothesis of the preferential attachment model, for each year of the available data, for every new node added to the network the degree of the entry node was recorded and collected in a histogram that approximates $\Pi(k)$ (each node is counted only once at its first entry). To smooth fluctuations, as in [12], the cumulative function $\mathrm{k}(k)$ is considered:

$$\mathrm{k}(k) = \int_0^k \Pi(x)dx \tag{1}$$

In case $\Pi(k) \propto k$ then $\mathrm{k}(k) \propto 2$, which reported in a log-log graph should be a straight line of slope 2.

Fig. 4 report $\mathrm{k}(k)$ for various years in the two FunkFeuer network and show clearly that there is no linear trend. Indeed, Fig. 5 confirms that for none of the years under analysis $\Pi(k)$ grows with k.

These results show that the two networks for which data is available (for the ninux network the monitoring period is too short) the growth model does not support the hypothesis of preferential attachment. As such, if the network keeps growing with the same model, coupled with the factual evidences explained in the next section, we do not expect the emergence of a scale-free behaviour.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

A. Interpretation of the results

Two features that influence the growth of a CN are: i) the limited range of wireless links ii) an upper bound on the number of incoming links.

Limited range: Wireless links are limited to a maximum length of about tens of kilometers and need to have line-of-sigh between the endpoints, while wired links do not have this limitation. Thus, a new node entering the network can not connect to any other node, and an existent node can acquire new links only from nodes placed at a distance smaller than the maximum range (which is not a fixed value and depends on a number of factors, such as the antenna type, the transmission power etc.). If the network grows in an urban area maintaining a constant density, hubs will be formed less likely than in a scale-free network.

Limited maximum node degree: A wireless node can be equipped with several physical radios, but more radios require more maintenance. Mounting tens of radios, cabling them, powering them, configuring them, is costly. While wireless ISPs use trellis and pay for the maintenance, a single person typically does not have the physical space, the resources and the time to install and maintain such a complex infrastructure. Thus, node degree can not grow indefinitely.

This result confirms and extends the analysis carried on portions of the Guifi network [7] that observed that some portions of Guifi did not show a scale-free behaviour. The authors suggest that this is true for networks that cover up to a certain geographical area and it is influenced by the degree of "planning" in the evolution of the network (planned or completely spontaneous). Another interpretation could be that Guifi, contrarily to the networks analysed in this paper is a real cooperative ISP, thus its mission is to bring Internet to the people. This probably leads to shorten as much as possible the path to the closest gateway, and a quasi-hierarchical network design is more suitable for this task. Instead, networks that have local connectivity as a goal may follow a different evolution path; more research is needed to formulate a sound interpretation.

IV. Socio-technical Analysis of Ninux

The following data is taken from network dumps collected in 2014 and extends the publication [5], the next two subsections expose the results and the third one jointly comments them. Direct interaction with the community was necessary to give a qualitative interaction of the quantitative results.

A. The Ownership of the ninux network

Fig. 6 presents the number of nodes possessed by the top 20 ninux participants, ordered by nodes owned, referring to a snapshot of the network in which the maximum number of nodes were present (140). Over a total of 78 owners, one user possess 17% of the nodes and the top five people own 31% of the nodes, top 13 people own roughly 50% of the nodes, 61 people own just one node. If we exclude the first individual (that we call P_{top}), the distribution is not particularly skewed, reflecting the fact that the number of owned nodes is generally limited by the number of physical locations in the city to which

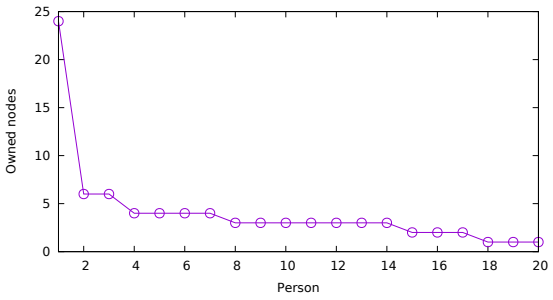


Fig. 6: The number of nodes per user in the ninux network, top 20 users.

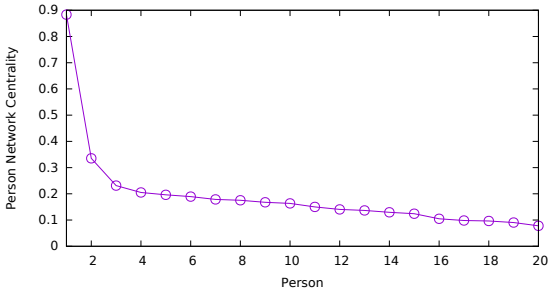


Fig. 7: The "person network centrality" for the participants to the ninux network, top 20 users.

the person has access (home, workplace, houses of relatives etc...). P_{top} owns 24 nodes and is not the owner of all the locations where the nodes are placed, he is simply a technically skilled person that very often offers his help to set up the network for newcomers. As a result, he appears to be the owner and the technical manager of the nodes.

Fig. 7 shows the group betweenness centrality computed on all the nodes owned by the same person. The group betweenness centrality is the fraction of shortest paths that pass through at least one node in the group. Formally, if the network graph is a weighted graph $G(V, E)$, and $P_{i,j} = \{v_i \dots v_j\}$ is the set of nodes that constitute the shortest path from node v_i to node v_j then the group centrality of a set of nodes $S = \{v_1 \dots v_n\} \subset V$ is given by:

$$B(S) = \frac{||\{P_{i,j}\ i, j \in (1 \dots |V|) \mid S \cap P_{i,j} \neq \emptyset\}||}{||\{P_{i,j}\ i, j \in (1 \dots |V|)\}||} \quad (2)$$

where $||\cdot||$ is the size of a set. The centrality metric is computed running Djikstra's algorithm on the weighted network topology, and, without information on the traffic matrix is the best estimation of the number of traffic flows that a group of nodes can intercept. Fig. 7 shows the "person network centrality", the ranked group centrality of the nodes owned by the same person and tells that P_{top} can potentially control almost 90% of the traffic flows.

Figs. 6 and 7 outline a peculiar feature of a CN. As long as people are not allowed to own nodes out of their properties,

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

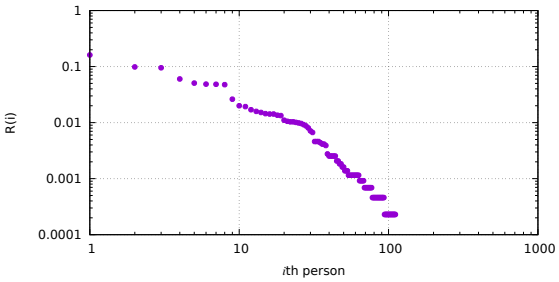


Fig. 8: The fraction of answered emails on the total in the mailing list.

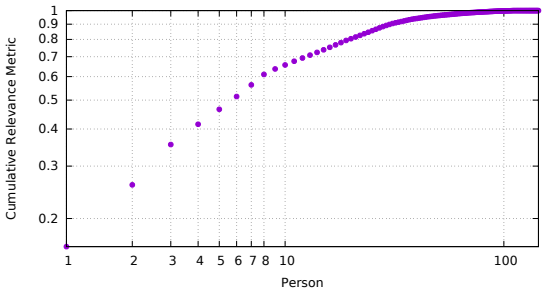


Fig. 9: The cumulative distribution of answered emails on the total in the mailing list.

the Wi-Fi range limitation does not allow a single person to be too central, and thus too critical for the network economy. This questions the peer-to-peer nature of the network in the case in which it is participated not only by individuals but also by associations, small business etc. that can be physically located in several places. In that case, the community must monitor the importance of those entities in order to avoid that any of them could become a single point of failure for the CN, as P_{top} is for ninux.

B. The ninux mailing list

The analysis of the mailing list messages helps understanding who are the individuals that lead the discussion inside the community. Two metrics defined in the literature have been chosen for this task [13]. The first is the normalized number of answered email per user: given a number X of total messages that reply to some other message, and being x_i the number of replies to a message sent by the ith person, $R(i) = \frac{x_i}{X}$ is the relevance metric shown in Fig. 8. This is a basic metric that assumes that people that receive a high number of replies are able to generate interesting discussion topics, thus are considered important in the community.

Fig. 8 shows that the relevance to the mailing list is not equally distributed among the participants, a very small number of people lead the discussion. The cumulative distribution in Fig. 9 shows that as less as 6 people receive 50% of the overall answers.

The second metric is the centrality of a person in the mailing list social graph. The social graph is an undirected graph $G(V, E)$ in which every node v_i is a person in the mailing list and there is an unweighted edge between two nodes v_i, v_j if person v_j ever answered to person v_i (or vice-versa). Mailing list centrality is computed on the social graph for v_i as in Eq. (2) when $S = \{v_i\}$. Betweenness centrality on mailing lists is used to understand who is able to make other people join the same discussion, so that he/she can facilitate the flow of information in the community. Again, Fig. 10 shows that there is a small number of people connecting all the other participants, and one in particular whose centrality is at least the double of the others.

Finally, Fig. 11 reports the percentage overlap on the two betweenness rankings from Fig. 7 and Fig. 10. The percentage

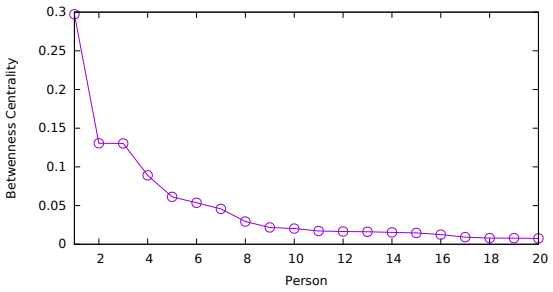


Fig. 10: The ranked centrality of the top 20 participants in the ninux mailing list.

overlap gives a measure of the correlation between the two rankings. Given a family of sets B_i and the respective ordering functions $o_i(v)$ on their elements, we call B_i^k the first k element of B_i ordered by $o_i(v)$: $B_i^k = \{v | v \in B_i, o_i(v) \leq k\}$. Given two sets B_1 and B_2 the percentage overlap $p(k)$ is a function of k that shows the percentage of elements present in both sets when considering only the first k elements:

$$p(k) = \frac{100}{k} \times ||B_1^k \cap B_2^k|| \quad (3)$$

Fig. 11 shows two fundamental points: the first is that P_{top}, the person that owns more nodes and has the highest person network centrality is the same one that has the highest mailing list centrality. The second is that excluding the top person the correlation is not extremely strong, $p(10) = 20\%$ and $p(20) = 35\%$.

C. Interpretation of the results

The distribution of the ownership, and thus the person centrality shows that, albeit the goal of the ninux community is to build a both technically and socially decentralized network, the results diverge from the goal. In 2014 one person in ninux managed a sufficient number of nodes to be able to control the network, and to represent a single point of failure. The same person, given his technical skills was a central person in the social network of the community, so he had an influential voice in the discussions. Indeed, direct discussion engaged

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

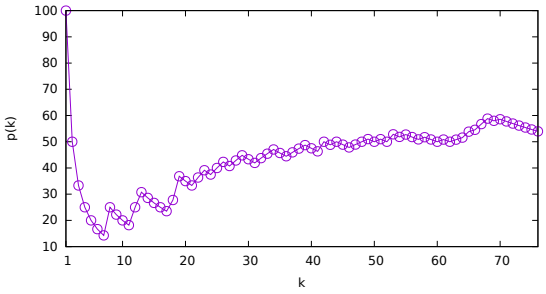


Fig. 11: The percentage overlap metric computed on the ranked mailing-list and group node centrality.

with people in the community revealed that this person left the community in 2015 and the nodes he managed started to fail and disconnect entire areas. At the time of writing the network is made of 87 nodes, 53 less than at its maximum expansion. In conclusion, the approach of the ninux community for the decentralization of the technical and social network was not successful since the network had a single point of failure represented by P_{top}.

However the situation changes excluding P_{top} from the analysis. Fig. 6 shows that the maximum number of owned nodes is generally capped by the amount of physical locations that the users have access to, which intrinsically limits the chances of some individuals to *take-over* the network. Also, even if the social network metrics show that the relevance of the participants to the mailing list is not evenly distributed (this is indeed pretty common in many mailing list [14]) the correlation between the most relevant node owners and the most relevant members of the mailing list is low. This means that people participate to the community in diverse ways, with the construction of new nodes or through rising discussion topics.

V. Conclusions

This paper analyses the evolution of two large CNs and produces a socio-technical analysis of a third one. The analysis shows that wireless network with a bottom-up, peer-to-peer organization do not match a preferential attachment growth model. The emergence of hubs is reduced so the topology is less dependent on hubs compared to a scale-free network. More research, and time, is needed to tell if CNs will represent with their growth a different network model, or they will converge to some other known models, and what are the effects on the applications that a CN can sustain.

If the *community networkers* want to pursue their goal of building decentralized socio-technical infrastructure they will have to monitor the social interactions in the community, to verify that no single person or small group of people can *take over* the network. Metrics such as network centrality and person centrality, together with social network analysis as used in this paper represent a starting point to develop monitoring instruments that will give to the community the "pulse" of the network and decide if the community is following a direction that best represents the community's collective goal. This is particularly important when the network evolves and it becomes the interconnection not only between private people but also between for-profit activities, as it happens in the Guifi network. In that case, the CN must monitor that a single entity does not grow large enough or central enough to become the effective "owner" of the network, reducing the control power of the community on the infrastructure.

References

[1] S. Jain and D. Agrawal, "Wireless community networks," *Computer*, vol. 36, no. 8, pp. 90–92, 2003.

[2] J. Sderberg, "Free Space Optics in the Czech Wireless Community: Shedding Some Light on the Role of Normativity for User-Initiated Innovations," *Science, Technology & Human Values*, vol. 36, no. 4, pp. 423–450, Jan. 2011.

[3] S. Crabu, F. Giovanella, L. Maccari, and P. Magaudda, "A Transdisciplinary Gaze on Wireless Community Networks," *TECNOSCIENZA: Italian Journal of Science & Technology Studies*, vol. 6, no. 2, pp. 113–134, Jan. 2016.

[4] L. Maccari, "An analysis of the Ninux wireless community network," in *The Second International Workshop on Community Networks and Bottom-up-Broadband (CNBuB)*, 2013.

[5] L. Maccari and R. L. Cigno, "A week in the life of three large wireless community networks," *Ad Hoc Networks*, vol. 24, Part B, no. 0, pp. 175 – 190, 2015.

[6] L. Cerda-Alabern, "On the topology characterization of Guifi.net," in *IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2012.

[7] D. Vega, R. Baig, L. Cerd-Alabern, E. Medina, R. Meseguer, and L. Navarro, "A technological overview of the guifi.net community network," *Computer Networks*, vol. 93, pp. 260–278, 2015.

[8] C. Barz, C. Fuchs, J. Kirchhoff, J. Niewiejska, and H. Rogge, "OLSRv2 for Community Networks: Using Directional Airtime Metric with external radios," *Computer Networks*, vol. 93, Part 2, pp. 324–341, Dec. 2015.

[9] L. Cerda-Alabern, A. Neumann, and L. Maccari, "Experimental Evaluation of BMX6 Routing Metrics in a 802.11an Wireless-Community Mesh Network," in *3rd International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug. 2015.

[10] R. Baig, R. Roca, L. Navarro, and F. Freitag, "Guifi.Net: A Network Infrastructure Commons," in *Proceedings of the Seventh International Conference on Information and Communication Technologies and Development ACMDev*. ACM, 2015.

[11] J. Kos, M. Milutinovic, and L. Cehovin, "nodewatcher: A substrate for growing your own community network," *Computer Networks*, vol. 93, pp. 279–296, 2015.

[12] H. Jeong, Z. Nda, and A.-L. Barabsi, "Measuring preferential attachment in evolving networks," *EPL (Europhysics Letters)*, vol. 61, no. 4, p. 567, 2003.

[13] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining Email Social Networks," in *Proceedings of the 2006 International Workshop on Mining Software Repositories*. ACM, 2006.

[14] S. L. Toral, M. R. Martnez-Torres, and F. Barrero, "Analysis of virtual communities supporting OSS projects using social network analysis," *Information and Software Technology*, vol. 52, no. 3, pp. 296–303, Mar. 2010.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

A flexible workload model based on roles of interactive users in social networks

Pablo Nicolás Terevinto, Ana Pont, José A. Gil and Josep Domenech
Universitat Politècnica de València (Spain)
Email: {pabtecha,jdomenech}@upvnet.upv.es, {apont,jagil}@disca.upv.es

***Abstract*—Uses, applications and functionalities of Online Social Network (OSN) are continuously changing and adapting to the new habits of users. The massive adoption of smart mobile devices and the appearance of new roles such as community managers have had a strong influence in the wide use of these networks among all sectors of population regardless of age, gender or socio-economic status. Consequently, the characterization and modeling of OSN user's behavior is a difficult task because their habits and activities change rapidly. This fact motivates us to propose a new OSN user's workload model based on the active roles that they play at any instant when navigating the application. Role-based workload is a flexible way to have a fresh characterization of users because they can adopt new roles, stop using others, or simply modify the way they change between them. Roles and their interrelations can be easily defined in our workload model and generator thus providing a useful tool for fine grain performance evaluation and testability studies.** [1]

***Index Terms*—Online Social Network, User characterization, workload, user role.**

1. Introduction

Social networking activity is a current global phenomenon that has surpassed all expectations. Statistics [1] show that more than 70% of all internet users are now active on social media, and experts project the number of social network users grow 12.5% each year. But maybe the most interesting point in this growth is the increasing presence of social networking in business and other activities different to leisure and entertainment, like for instance education, research and more timidly for e-governance.

But despite its penetration and massive use, the technical paradox in this kind of applications is the lack of tools, workload models and testbeds for performance evaluation and testability studies. This fact makes difficult, among others, provisioning hardware and software resources in an efficient and appropriate way according to the functional use and workload previsions. As with any other web-based application, the complexity of characterizing a wide spectrum of user's behaviors and the continuous emergence of applications that change user's habits makes it extremely difficult to get representative workloads for these important studies. This fact is even more relevant due to the intrinsic difficulty for many research communities to get real traces which are usually owned and kept by big companies.

In a previous work [2], we proposed the Dynamic Web Workload Model (DWEB). This model makes it possible to characterize and reproduce the behavior of web users which is usually a difficult task due to the continuous interaction between them and the offered content. To do this, DWEB introduced two concepts that permit to consider different levels of user interactions. First, the user navigation concept allows us to represent dynamic reactions of users when they interact with web content and services. These reactions modify the user's response according to the content provided by the OSN server or other parameters, as for instance, response time or quality of service. This feature permits to create interactive users. Second, the user roles concept defines the different behaviors of users according to the characteristics of the visited site, their personal goals and their active involvement. By implementing these two concepts in our workload generator we can mimic the behavior of the actual web users' community.

Departing from the versatility offered by DWEB, this paper proposes a workload model based on roles representing a variety of OSN user's profiles. In this paper we propose a workload model based on roles. We present three different user's roles which have been currently identified in different OSN (Facebook, Twitter, LinkedIn, MySpace, Pinterest, Tuenti). These roles are representative of the most common user's profiles in these kind of networks and can be easily identified in one degree or another in all of them. Nevertheless, the activities that can be performed in each role can be different according to the services offered by each application. User's roles define a level of abstraction that allows us to create a flexible model which can be adapted to any kind of OSN by modifying the activities performed by each role.

In summary, the main contributions of this work are: i) a flexible and adaptable OSN workload model based on the main current user's roles, ii) a generation of a dynamic workload that accurately reproduces interactive OSN user.

The remainder of this paper is organized as follows: in Section 2 we present the roles of our model and describe the activities of an OSN which are later used to create the

1. This work has been partially supported by the Spanish Ministry Economy and Competitiveness under grant TIN-2013-43913-R

ISBN 978-3-901882-83-8 © 2016 IFIP

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

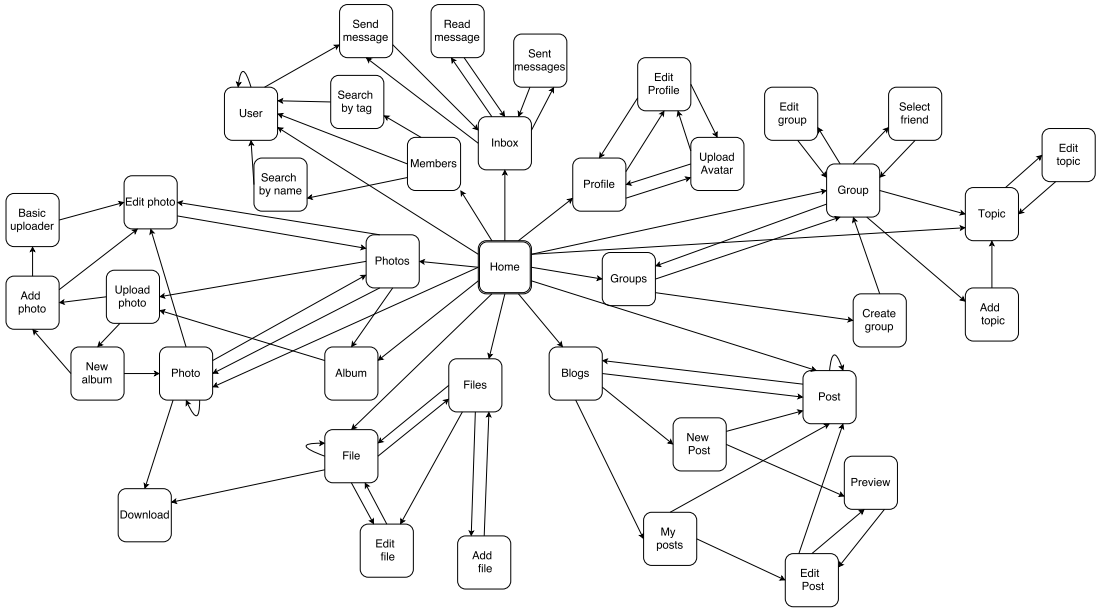


Figure 1. Navigational graph based on the OSN interface.

workload model. Then, the model is used to generate the workload as Section 3 describes. Later in Section 4 we provide a comparison of our work with the most related works that can be found in the open literature. Finally, we provide the main conclusions of our work and give some ideas about our future work.

2. Workload modeling based on user roles

In this section we present a workload model which is based on the concept of role implemented in DWEB. For this, we present the typical roles identified and an example of this model for an specific OSN. First of all, for sake of clarity, we define the nomenclature used in this work.

- **Role:** defines the user's behavior while interacts with the application. Roles define the active involvement of users and produce a set of sequences of a activities that can be done in parallel and/or sequentially by each user.
- **Activity:** is the sequence of actions performed by the user aimed at achieve a simple goal. Examples of activities are: uploading a file, posting a blog or commenting a photo. Many current OSN offer similar activities, but the way they are made can differ in each one. This is because the sequence of actions involved is different.
- **Action:** each of the interactions between the user agent and the application (OSN). They are originated from the interrelation between user and the application's interface. Actions can trigger the request of a new page or the change of state of the current one. However, not all user's interactions produce an action. For instance, filling up a form may not produce an action, while sending it does. Actions are intrinsic to the application. Between the successive actions of an activity usually it elapses time, which is the latency between an action and the user's reaction, as a consequence of the previous ones (think time).

2.1. User's roles definition

An accurate definition of roles is important to later develop a flexible workload model that: i) can represent the activities done by users when they use this type of applications, ii) permits to conduct fine grain evaluation studies. The user roles that we propose in this paper are representative of the current behaviors of any individual when they participate in a OSN. Those roles are: user, generator user, and reader user. These profiles have been also identified in [3] after analyzing real traces from a blogsphere. This observation is also consistent with the user characterization presented in [4] and [5]. And finally, some internet monitoring websites [6] and [7] provide similar inputs about OSN user's active participation.

- **Social user:** this type of users browse the OSN, but also interact with other users and generate new content in the OSN.
- **Generator user:** content generator users are those whose main purpose is to update and upload new content to the social network and respond to other user's comments and messages. This is the typical

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

role for a social media manager or community manager for example.

- **Reader user:** users which generally log in to check the updates in the OSN. Reader user do not interact with other users or comment any content. They only browse the OSN.

2.2. Activities for an OSN

Although roles are quite independent from the type of OSN, the activities of each role are not. The activities and the actions that implement the activities are dependent of the services offered by the OSN. For this reason and in order to present an example model, an online social network powered by the open source social networking engine Elgg was used. The Elgg 1.8 version was selected with the plugins for the most common activities which users usually undertake while navigating OSN.

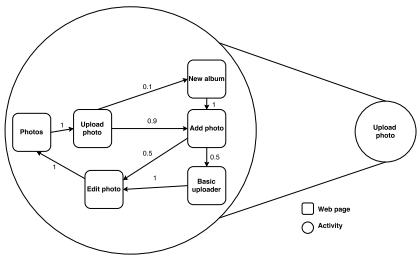


Figure 2. Grouping several actions into the activity *upload photo*

Our first step was an analysis of the OSN interface and the possible transitions between the different pages. Here we identified the starting point of the web site, the Home page. It provides a timeline with the latest visible changes occurred on the site and therefore, offers direct links to specific content such as photos, files, posts, albums, groups or members. Also, the OSN interface has fixed links to each of the types of content mentioned before. This analysis resulted in the navigational graph shown in Figure 1 where each square node represents a single page of the OSN. This graph shows the transitions present in each of the OSN web pages without considering the transitions between different types of content which are always present in every page. This omission aims to reduce the complexity of the graph. Although those transitions have been omitted in the figure, they will be taken into consideration in the following steps of the modeling process.

Using this graph we identify the activities that a user can perform in our OSN.

As an example, Figure 2 shows how we have grouped the different necessary actions to upload a photo to the social network into the activity *upload photo*. Each square node represents a web page while activities are represented with circles. Besides the transitions between pages, an activity also includes actions which provide interactivity. For instance, after each transition we can modify the user's response time or the length of the answer according to the amount of data of the server response. Responses with more information usually require more time for the user to process them. Quality of service could also affect an action because the user may decide to stop browsing after several unsuccessful connections to the server or due to high latency. These factors can influence the probabilities of transition, changing the activity of closing the session.

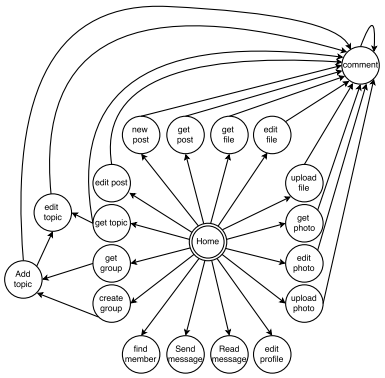


Figure 3. Navigation graph for a social user.

2.3. Linking activities to user's roles

At this point, we took into consideration the definitions of roles aforementioned and selected those activities which better suit each role. Additionally, the user's roles are finally defined by giving probabilities to the transition between activities

Social role model. A social user spends most of the time browsing the social network, but eventually interacts with other users or generates new content to share with the rest of users.

In our model, we considered the results presented in [4] stating that: i) 92% of the total amount of workload consist in browsing content, ii) users tend to repeat the same activity around 67% of the time, iii) users tend to do related activities instead of doing unrelated ones.

The navigational graph for the social role is the result of grouping all the activities described in the previous section. Figure 3 represents the initial Home page and the transitions to the different activities. For sake of clarity, the activities not accessible from the starting page where added to the graph but the transitions between them have been omitted. The complete representation of transitions is presented in Table 1. It represents the transitions between activities and the probabilities for each transition with values represented in percentage. These values and the ones presented in other tables are merely an example, although they are enough consistent with users navigations. Most of the time spent is browsing the site and mainly repeating the same activity. Also, changes between activities tend to occur to related activities.

Generator role model. Content generator users only perform activities which produce new content or modify

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

TABLE 1. PROBABILITIES (%) OF TRANSITIONS BETWEEN ACTIVITIES FOR A SOCIAL ROLE.

	Home	Get post	Get file	Get photo	Get group	Get topic	New post	Upload file	Upload photo	Create group	Add topic	Edit post	Edit file	Edit photo	Edit topic	Comment	Read message	Send message	Find member	Edit profile
Home	-	18	18	18	18	18	1	1	1	1	-	1	1	1	1	-	0.5	0.5	0.5	0.5
Get post	3	66	4	4	4	4	2	-	-	-	-	1	-	-	-	10	0.5	0.5	0.5	0.5
Get file	3	4	66	4	4	4	-	2	-	-	-	-	1	-	-	10	0.5	0.5	0.5	0.5
Get photo	3	4	4	66	4	4	-	-	2	-	-	-	-	1	-	10	0.5	0.5	0.5	0.5
Get group	13	19	19	19	4	19	1	1	1	1	1	-	-	-	-	-	0.5	0.5	0.5	0.5
Get topic	3	4	4	4	4	66	-	-	-	-	2	-	-	-	1	10	0.5	0.5	0.5	0.5
New post	12	50	5	5	5	5	3	-	-	-	-	3	-	-	-	10	0.5	0.5	0.5	0.5
Upload file	12	5	50	5	5	5	-	3	-	-	-	-	3	-	-	10	0.5	0.5	0.5	0.5
Upload photo	12	5	5	51	5	5	-	-	-	-	-	-	-	5	-	10	0.5	0.5	0.5	0.5
Create group	-	-	-	-	-	-	25	25	25	-	25	-	-	-	-	-	-	-	-	-
Add topic	12	5	5	5	5	50	-	-	-	-	3	-	-	-	3	10	0.5	0.5	0.5	0.5
Edit post	59	10	2	2	2	2	1	1	1	1	1	1	-	-	-	15	0.5	0.5	0.5	0.5
Edit file	59	2	10	2	2	2	1	1	1	1	1	-	1	-	-	15	0.5	0.5	0.5	0.5
Edit photo	59	2	2	10	2	2	1	1	1	1	1	-	-	1	-	15	0.5	0.5	0.5	0.5
Edit topic	59	2	2	2	2	10	1	1	1	1	1	-	-	-	1	15	0.5	0.5	0.5	0.5
Comment	8	16	16	16	16	16	1	1	1	1	1	1	1	1	1	1	0.5	0.5	0.5	0.5
Read message	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	15	65	-	-
Send message	80	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	20	-	-	-
Find member	-	19	18	19	18	18	-		-	-	-	-	-	-	-	-	-	8	-	-
Edit profile	3.5	18	18	19	18	18	1	1	1	1	1	-	-	-	-	-	0.5	0.5	0.5	-

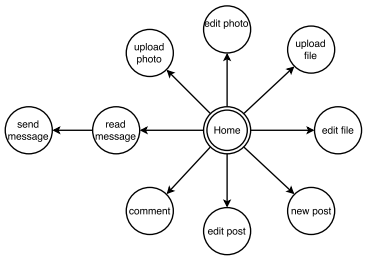


Figure 4. Navigation graph for a generator user.

existing one. We also consider that these users read and answer private messages from other users as well as they answer comments from other users on the generated content.

Figure 4 shows the navigation graph including only the transitions from the initial node (Home) to each activity. To reduce the complexity of the graph given the high density of connections between nodes, the remaining transitions are shown at Table 2 where each probability is expressed in percentage. As this type of users do not tend to browse content, the navigations had to be readjusted by giving more weight to the actions that create new content rather than those that modify existing ones. Also, we have considered that this kind of user uses the private messages mainly to answer other users. Therefore the activity *Send message* only follows a *read message* first.

Reader role model. The only interaction of this type of users with the OSN is requesting content. They do not interact with other users nor generate new content to contribute with the OSN growth. With this in mind, we have selected the activities which allowed the user to browse the site without creating new content. Figure 5 shows the graph

TABLE 2. PROBABILITIES (%) OF TRANSITIONS BETWEEN ACTIVITIES FOR A GENERATOR ROLE.

	Home	New post	Upload file	Upload photo	Edit post	Edit file	Edit photo	Comment	Read message	Send message
Home	-	18	18	18	2	2	2	20	20	-
New post	5	17	17	17	4	-	-	30	10	-
Upload file	5	17	17	17	-	4	-	30	10	-
Upload photo	5	17	17	17	-	-	4	30	10	-
Edit post	23	17	17	17	1	-	-	15	10	-
Edit file	23	17	17	17	-	1	-	15	10	-
Edit photo	23	17	17	17	-	-	1	15	10	-
Comment	20	16	16	16	-	-	-	22	10	-
Read message	10	13	13	14	-	-	-	-	15	35
Send message	45	15	15	15	-	-	-	-	10	-

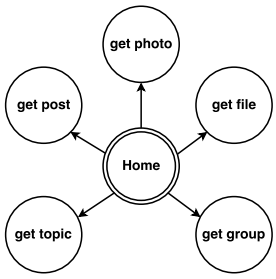


Figure 5. Navigation graph for a reader user.

with the Home page as the starting point, and the activities from the initial node.

Table 3 represents the transitions between activities and the probabilities for each transition with percentage values. Every activity has a similar behavior, except for *Get group* which tends to get topics, posts, photos or files of that

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

specific group. Also here it can be appreciated that the user tends to repeat the same activity.

TABLE 3. PROBABILITIES (%) OF TRANSITIONS BETWEEN ACTIVITIES FOR A READER ROLE.

	Home	Get post	Get file	Get photo	Get group	Get topic
Home	-	20	20	20	20	20
Get post	2	70	7	7	7	7
Get file	2	7	70	7	7	7
Get photo	2	7	7	70	7	7
Get group	-	25	25	25	-	25
Get topic	2	7	7	7	7	70

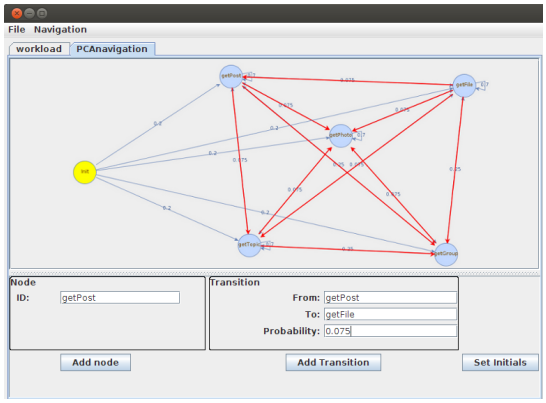


Figure 6. Reader user's navigation graph implemented with LoadG.

3. Workload generation

This characterization of user's behaviors in roles, and the proposed model are addressed to finally generate a representative workload for testing applications and for any type of fine grain performance evaluation studies. To this end, we implement a workload with interactive users using GUERNICA and LoadG. GUERNICA [8] is a web workload generator which allows the user to define interactive web workloads based on the concept of DWEB previously mentioned and considering also current browsers facilities. LoadG is a graphical configuration tool which provides a user-friendly interface to create the navigation and workload XML files to configure GUERNICA. LoadG helps to easily design the models by graphically adding new nodes and interconnect nodes assigning probabilities of transitions between them. Also, each node can be edited and the user can add one or more actions associated with the node. In this case, we define each activity as a node as shown in the previous sections and each node contains the different actions necessary to achieve the goal of the activity. Once the model is implemented, LoadG allows the user to export the model in a XML format which GUERNICA can interpret. At this point, the generator permits to define conditional transitions among activities, considering, for instance, the results of previous actions, QoS parameters or the time of day. Figure 6 is a screenshot of the reader user's model implemented with LoadG. This interface offers different colors and line thickness to distinguish between a single transition and multiple transitions.

4. Related Work

Unlike other computational workloads, web-based application ones involve people who interact and are sensitive to the offered content. That makes it difficult to have available representative workloads for performance evaluation or testability studies. In order to provide detailed and realistic workloads for web-based applications, the faithful characterization of users behaviors is a fundamental pillar. One of the earliest attempts to characterize in detail web users behavior can be found in [9] where the intrinsic characteristics of web workloads were settled and the importance of considering the user interactivity was pointed out. For the case of e-commerce sites, this work identified the main user transactions which supposed a workload characterization based on user's activities according to our nomenclature.

This user model was later extended in [10] to capture application inter-request and data dependencies in order to consider a certain degree of interactivity in the workload. More recently, [11] characterized and modeled generic web user's navigations that include current browsers facilities such as the use of back button or opening new tabs, and they also represented the user dynamic interaction with the provided contents or according to QoS parameters.

With the growing and wide penetration of social network applications, it becomes necessary to define specifics web workload models for them. A recent work [11] presents an interesting survey where several approaches to characterize OSN users are explored. They classified these attempts according to the point of view of connection, interaction, traffic activity, mobile social behavior, and malicious behavior. This work also focuses on the importance of understanding user's behavior both for Internet and Applications service providers. Also a better knowledge of user's behavior can help for enhancing user experience.

The most related work to ours [3] defines the workload for a blogspace. In this work, a similar idea to ours for identifying user's roles was made for the first time. They describe profiles like "blurker" and "commenter" and also distinguish between write session and read session associated to those profiles. But, this work only considers the scope of the blogosphere which is only a specific case of social network.

In [4], they characterize users behavior in online social networks by collecting and analyzing data obtained from a representative social network aggregator. While their model relies on user activities and the transitions between them, our work, provides a higher level of abstraction, by identifying the main user's roles in OSN according to the major functionalities and characteristics of these applications [12].

Other authors have focused on characterizing specific

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

user's actions or roles. For instance, the work presented in [13] models the user posting behavior on social media according to the influence of content factors but it does not consider other user roles which are also of huge interest.

Another approach can be found in [14], where a normal OSN user's behavior is characterized and modeled in order to identify significant deviations aimed at detecting anomalous or malicious activities. Authors use statistical techniques applied to user pattern accesses, such as the number of likes at day. This level of characterization does not permit to easily identify user's roles or specifics profiles.

An interesting work to assist the workload generation process for OSN is [15]. They characterize the workload at the level of user navigation, identifying sessions, subsessions within a session, session durations, active and inactive times, inter arrival time, bytes per session, and they pay also attention to popularity. Nevertheless, some data should be reviewed for updating the results to the current context.

Aimed at detecting user's intentions and preferences for efficient recommendations system design, the work presented in [16] also analyzes user behaviors in social media systems considering temporal context. This work focuses only on the creation of more accurate actions and recommendations, but does not consider user's profiles

In general, pro-active users of social networks not only focus on one type of them but they are also active members of others. Therefore, to understand how users distribute their activities across different sites is also of interest. In these vein [17] investigates the relationship between user's patterns of two well-known social sites to better understand this phenomenon.

5. Conclusions and future work

In conclusion, in this paper we have presented a new method for designing a flexible model to represent the interactive behavior of OSN users.

Our model is based on the typical roles that active users play currently in these networks. Roles present a level of abstraction when modeling the workload that permit to offer a flexible and fresh characterization. Consequently, it can be easily adapted to the ever-changing environment of these applications, where new trends, functionalities and technologies are appearing constantly.

By implementing the resulting model in our workload generator (GUERNICA) we can reproduce in an accurate way the interactive actions performed by OSN users when navigating. Our generator allows conditional user navigations depending on the content offered or other type of parameters related, for instance, to QoS metrics. The combination of GUERNICA and LoadG, a powerful and friendly tool to graphically define and edit web based workloads, permits to generate detailed workloads for performance evaluation and testability studies.

The model presented in this work has been applied to the specific case of Elgg, an open source OSN that includes the typical functionalities of these type of networks. At the present, Elgg has been conveniently filled out with contents to serve as back end for a future benchmark aimed, among others, at evaluating OSN.

References

[1] I. W. Stats, "Usage and statistics," http://www.internetworldstats.com/.

[2] R. Pea-Ortiz, J. Sahuquillo, A.Pont, and J. A. Gil, "Dweb model: Representing web 2.0 dynamism," *Computer Communications*, vol. 32, no. 6, pp. 1118 – 1128, 2009.

[3] F. Duarte, B. Mattos, J. Almeida, V. Almeida, M. Curiel, and A. Bestavros, "Hierarchical characterization and generation of blogosphere workloads," 2006.

[4] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 49–62.

[5] M. Maia, J. Almeida, and V. Almeida, "Identifying user behavior in online social networks," in *Proceedings of the 1st Workshop on Social Network Systems*, ser. SocialNets '08. New York, NY, USA: ACM, 2008, pp. 1–6.

[6] Statista, "Social networks usage in spain," http://www.statista.com/study/31830/social-networks-usage-statista-dossier/.

[7] S. M. Examiner, "Facebook and twitter user behavior changes: New research," http://www.internetworldstats.com/.

[8] R. Pea-Ortiz, J. Sahuquillo, A. Pont, and J. A. Gil, "Generating dynamic workload for web performance evaluation," *XXI Jornadas de Paralelismo (JJPP'10)*, pp. 711–718, 2010.

[9] D. A. Menasce and V. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[10] M. Shams, D. Krishnamurthy, and B. Far, "A model-based approach for testing the performance of web applications," in *Proceedings of the 3rd International Workshop on Software Quality Assurance*, ser. SOQUA '06. New York, NY, USA: ACM, 2006, pp. 54–61.

[11] L. Jin, Y. Chen, T. Wang, P. Hui, and A. Vasilakos, "Understanding user behavior in online social networks: a survey," *Communications Magazine, IEEE*, vol. 51, no. 9, pp. 144–150, September 2013.

[12] J. Heidemann, M. Klier, and F. Probst, "Online social networks: A survey of a global phenomenon," *Computer Networks*, vol. 56, no. 18, pp. 3866 – 3878, 2012, the {WEB} we live in.

[13] Z. Xu, Y. Zhang, Y. Wu, and Q. Yang, "Modeling user posting behavior on social media," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '12. New York, NY, USA: ACM, 2012, pp. 545–554.

[14] B. Viswanath, M. A. Bashir, M. Crovella, S. Guha, K. P. Gummadi, B. Krishnamurthy, and A. Mislove, "Towards detecting anomalous user behavior in online social networks," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 223–238.

[15] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding online social network usage from a network perspective," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 35–48.

[16] H. Yin, B. Cui, L. Chen, Z. Hu, and Z. Huang, "A temporal context-aware model for user behavior modeling in social media systems," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 1543–1554.

[17] R. Ottoni, D. B. Las Casas, J. P. Pesce, W. Meira Jr, C. Wilson, A. Mislove, and V. Almeida, "Of pins and tweets: Investigating how users behave across image-and text-based social networks." in *ICWSM*, 2014.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

On Improving Tie Strength Estimates by Aggregating Multiple Communication Channels

Narges Yousefnezhad
Aalto University
narges.yousefnezhad@aalto.fi

Marcin Nagy
Aalto University
marcin.nagy@aalto.fi

N.Asokan
Aalto University & University of Helsinki
asokan@acm.org

***Abstract*—The degree of closeness in a relationship is characterized as *tie strength*. Estimates of tie strength can be useful in many contexts, including as a parameter in access control policies or social context based services. Several papers have proposed how tie strength can be estimated by quantifying interactions in different individual communication channels such as online social networks, phone communication and face-to-face encounters. It has been conjectured by Wiese et al. [1] that considering only a single communication channel may not lead to accurate estimates of tie strengths. In this paper, we explore this conjecture by examining whether the combination of co-location events and mobile communication data can lead to better tie strength estimations than considering each channel individually. Surprisingly, our results indicate that the conjecture may not be true, but further analysis with more extensive datasets is needed to confirm the result.**

I. Introduction

Tie strength is a notion used by social scientists to represent the degree of closeness in a relationship between two people [2]. The ability to accurately estimate tie strengths among people can lead to new services or improvement of existing ones. For instance, travellers and commuters can use tie strength estimation to decide if they want to share a ride with a stranger [3]. Similarly, people can decide to share their mobile data connection with close friends, specified as the list of their contacts with tie strength values above some threshold [4]. Generally, estimation of tie strength has many important applications in user-controlled online identity authentication [5], consumer behaviour prediction systems [6], recommendation services [7] and reputation services [8].

Prior research on tie strength estimation has largely focused on using input data from a single *communication channel*. We define a communication channel as any medium that can be used for exchange of information between people. Most studies estimate tie strength based on three communication channels: online social network (OSN) interactions [9], [10], traditional telecommunication such as calls and text messages [11], [12], [13] and interactions based on physical proximity [14], [15], [16].

Intuitively, information about interactions in *different* communication channels is likely to be a more accurate predictor for tie strength values. Several previous works [2], [17], [18], [1] have touched on this topic. Want et al. [17] conducted user studies to understand how well interactions over different communication channels correlate with closeness of friendship. Hritsova et al. [18] showed that people who use multiple types of channels for communicating with each other are more likely to have higher tie strengths between them. Wiese et al. [1] showed that when only one communication channel (interactions via telecommunication networks) is considered, the resulting tie strength estimates may be incorrect. They further concluded that combining information from different communication channels can lead to more accurate estimation of tie strength values. None of above investigated concrete tie-strength computation techniques that use multiple communication channels to confirm whether the conjecture is correct.

In this paper, we explore this question by using machine learning classifiers to predict tie strengths in order to evaluate whether combination of data from different communication channels leads to a better prediction accuracy. We use an existing dataset [19]. Our results indicate that while this conjecture may be true, it cannot be claimed with statistical significance. We therefore conclude that a more extensive dataset would be needed in order to resolve this question more definitively.

ISBN 978-3-901882-83-8 © 2016 IFIP

II. Background

Tie strength was introduced by Granovetter in 1973 [20]. He defines strength of a tie between two people in the social network as a combination of four factors: the amount of *time* people spend with each other, *emotional intensity*, *intimacy* (mutual confiding), and *reciprocal services* that characterize the tie. Furthermore, he also divides ties into two classes: *weak ties* that link acquaintances and *strong ties* that are formed between people trusting each other. There is a lot of published prior work on tie strength estimation with a particular focus on assigning binary values (strong or weak) to ties [17], [5], [21] and labelling them [22], [8].

We now present a brief summary of recent works on tie strength estimation using a single communication channel and discuss the shortcomings of relying only on a single channel.

A. Tie strength in a single communication channel

We consider three types of communication channels: online social networks (OSN), mobile communication networks and physical proximity.

Tie strength via OSN interactions. People using OSNs often have a very large number of contacts. Although most

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

OSNs provide the functionality of assigning social contacts to specific sets (e.g., family, acquaintances, etc.) that reflect various degrees of closeness, people usually do not bother to take advantage of such functionality. To automate this process, Gilbert et al. [23] and Spiliotopoulos et al. [24] proposed using tie strength estimation methods based on a linear combination of factors described by Granovetter as well as emotional support and social distance. Arnaboldi et al. [9] defined 19 Facebook features, found their correlation to tie strength and presented two linear models for tie strength estimation. They concluded that recency of contact between people has the highest impact on tie strength. Jones et al. [10] extracted 14 features and developed a logistic regression model to check importance of extracted features. They, however, showed that interaction frequency is the most important feature in determining tie strength.

Tie strength via mobile communication network interactions. Before OSNs became hugely popular, tie strength estimation research largely concentrated on interactions via (mobile) communication networks. Onnela et al. [11] examined social communication patterns based on phone calls and SMSes. They applied duration of calls for tie strength estimation to show the existence of a relationship between tie strength and local social network structure. Zhang and Dantu [12] presented an affinity model for predicting social ties relying on communication logs. Eagle et al. [25] analysed status of friendship based on mobile phone record data.

Tie strength via interactions in physical proximity. Tie strength can also be estimated based on co-location events (proximity interactions) between two people. Crandel et al. [15] found that high number of physical proximity interactions between two people corresponds to the higher probability of a strong tie between them. Bilogrevic et al. [16] used the notion of an *encounter* (defined as co-presence of two people for a sufficiently long duration) for estimating tie strength. Sekara et al. [26] presented tie strength estimation based on proximity as determined by Bluetooth encounters.

B. Shortcomings of using a single communication channel

Although tie strength estimation based on a single communication channel gives a fairly accurate results, applications like access control can benefit from increased accuracy. For instance, tie strength estimation based solely on physical proximity interactions is affected by the *familiar stranger* [27] phenomenon, which can causes the strength of some ties to be overestimated. Similarly, ties between people that are not usually co-located (e.g., in long-distance relationships) will be underestimated. Wiese et al. [1] showed that tie strength estimation based only on mobile communication interactions causes about 50% of strong ties to be incorrectly classified as weak ties. They concluded that there is a strong motivation for building tie strength estimation methods that connect input data from multiple communication channels.

III. Multi communication channel Tie Strength

We now discuss our multi communication channel tie strength estimation model. We begin with a description of the dataset we worked with, including an overview of features we use in our model. Later, we describe the three tie strength estimation models.

A. Dataset

We have two main requirements for the dataset to fulfill: (1) presence of at least two different communication channels and (2) ground truth about the tie strength between pairs of people. We chose the *MIT Social Evolution* dataset [19] which contains traces from everyday life of 80 students living in the dormitory on the MIT campus. The dataset includes two communication channels: physical proximity (based on Bluetooth scans) and mobile communication network interactions (logs of phone calls and SMSes). The dataset covers nine months beginning from October 2008.

Data volume. The dataset contains 372 instances (pairs of people) with mobile communication interactions and 4770 instances with physical co-presence. 367 instances have interactions in both communication channels, and can thus be used for evaluation of tie strength estimation in a multi communication channel model.

Ground truth. During the data collection campaign, participants were asked which other participants they consider to be *close friends* with. Thus, if a participant has indicated that he/she is a close friend of another participant, we recognize their tie as strong. Otherwise, we consider their tie as weak. Overall, the ground truth is skewed, as only 668 pairs out of 4770 total pairs of users in both interactions indicated strong tie.

B. Multi-channel Tie Strength Model

Definition. We define the multi communication channel tie strength as a tie strength between two individuals that includes communication features coming from multiple communication channels.

Figure 1 illustrates difference in tie strength estimation between single and multiple communication channel approaches. If only mobile communication channel is involved, tie strength between Bob and John cannot be estimated. Similarly, tie strength between Bob and Alice cannot be calculated if only physical proximity channel is considered. However, by aggregation of data from multiple communication channels, all possible tie strengths between them can be estimated. Furthermore, tie between Alice and John can be estimated more accurately, as it includes data coming from both communication channels.

Feature Extraction. Recall from II-A that *contact duration*, *contact frequency*, and *recency of contacts* are considered as the most important features for tie strength estimation. We use them as the basic features both in mobile communication as well as physical proximity channel. In addition, we derive several new features which are based on distribution of these basic features (e.g., percentiles of call duration).

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

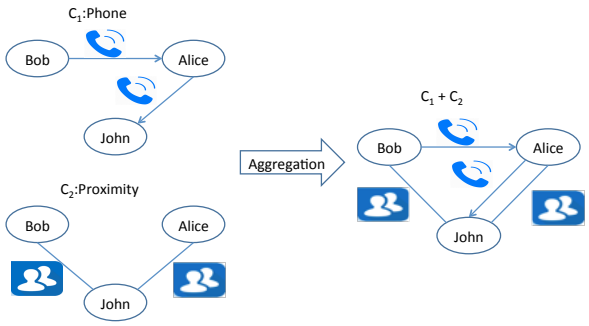


Fig. 1: Social network based on multiple communication channels.

Having extracted the features for mobile communication and physical proximity channels, we build two models, namely the **Mobile Communication-only** model and the **Proximity-only** model based on the features from respected channels. Finally, we create the new model (which we call **Aggregation**) by combining the features from the both channels.

Model description.

- ***Mobile Communication-only.*** The most important mobile communication features are identical to the top five features used by [1]. Furthermore, we define also additional 11 features describing various percentile levels of inter-communication times (i.e., time intervals between subsequent communication between two people). In total, the model includes 16 mobile communication features (see Table II for details).
  Furthermore, we assume mobile communication model follows *unidirectional* character of user interactions. This is motivated by intuitively different impact of incoming and outgoing calls and SMSes on tie strength estimation. For instance, if Alice calls Bob, it can be implied that she is interested in him, but the reverse interest cannot be proven.
- ***Proximity-only.*** Following Bilogrevic et al. [16], we use *encounter* as the primary feature. Based on it, we derive a total of 17 proximity features ranging from simple total encounter counts and mean encounter duration to more sophisticated percentile based features describing distributions of encounter and inter-encounter durations (see Table II for details).
  Unlike in the mobile communication-only model, we assume user interactions in the Proximity-only model to be *bidirectional*. This is motivated by assumption of mutual interest of two people during a co-presence event.
- ***Aggregation.*** This model includes both the Mobile Communication-only and the Proximity-only features. In total, the model has 33 features.
  Since the aggregation model is a combination of the Mobile communication-only and the Proximity model, for which notions of interactions are unidirectional and bidirectional respectively, we assume also unidirectional notion of user interactions in this model.

IV. Evaluation

This section presents accuracy of tie estimation achieved by our models. We begin with description of the dataset preparation for our evaluation. After that we describe how we checked that the dataset includes similar characteristics to the dataset used by Wiese et al. [1]. Finally, we evaluate the Aggregation model and compare accuracy of tie strength estimation with the Mobile Communication-only and the Proximity-only models.

A. Dataset preparation

Preprocessing. Recall from Section III that the Proximity-only model assumes user interactions to be bidirectional. Unfortunately results of Bluetooth scans may be asymmetrical (e.g., if Bob and Alice are co-located, only Bob's device discovers Alice presence, while her device does not discover him). This can happen for two reasons: (1) strong interference coming from neighbouring devices makes some devices unable to respond to Bluetooth discovery inquiry and (2) two parties are separated by some distance/obstacle (and in fact not co-present) and consequently Bluetooth signal between them is very weak and detectable only by one party. To compensate for this problem, we assumed the former reason and manually updated proximity data, as if the parties have been able to mutually discover themselves. On the other hand, we also checked that removal of asymmetric Bluetooth scan results (i.e., assuming the latter reason) does not change the accuracy of tie strength estimation results.

Normalization. Due to wide range of values that the features take, we apply the normalization of features with the range of 0 and 1.

B. Dataset validation

Wiese et al. [1] concluded that reliance only on the mobile communication network channel produces many errors in tie strength estimations and needs to be updated with more communication channels. Furthermore, as the dataset presented in their evaluation shows specific characteristics, we validate that our dataset exhibits similar characteristics.

They evaluated accuracy of tie strength estimation for three different input data. The first set of data (all) contains tie strength estimations between users and randomly chosen 70 contacts from the phone book and the Facebook contacts. The second set ($contactlist$) includes only contacts contained in user's phone book. Finally, the third set ($somecomm$) includes only contacts to which user has made at least 1 phone call or exchanged 1 SMS. Their results show a clear trend that precision of weak ties classification decreases if there are less contacts with whom user does not actively communicate (see

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Table I for details). The reason behind this performance drop is the fact that most of the ties without active communication are weak ones, thus are easier to classify.

Since our dataset does not contain the notion of the Facebook contacts and the phone book, we must generate a *simulated phone book* which includes all users contained in the dataset. We have following inputs:

- $fullbook$**:** includes all possible pairs of dataset users. Our dataset contains 80 users, so there are $80*79 = 6320$ pairs in total. If there is a recorded communication between a pair of users, we assign mobile communication-only features for them, otherwise we set values for all features to zero.
- $someEn1$**:** includes all possible pairs of users in the dataset with at least one recorded co-location event. It has 4403 pairs in total.
- $someEn10$**:** includes all possible pairs of users in the dataset with at least 10 recorded co-location events. It has 3893 pairs in total.

We validated that our dataset shows similar trends to Wiese et al.'s by evaluating our inputs using the Weka Toolkit [28]. We balanced ground truth using Synthetic Minority Oversampling Technique (SMOTE) [29] and used implementation of the Sequential Minimal Optimization (SMO) [30] with 10-fold cross-validation as the classifier. As the strong tie in our dataset is indicated by "close friend" label, for comparison we chose $2 - mediumstrong$ class condition from Wiese et al. which classifies tie strength into two classes (strong-medium ties and weak ties). Our dataset exhibits similar performance drop trend as reported by Wiese et al., thus it can be used for evaluation of multi communication channel tie strength estimation (see Table I for details).

TABLE I: Comparison of performance drop in precision of weak ties classification for Wiese et al. [1] and our dataset.

		Strong ties		Weak ties	
		Precision	**Recall**	**Precision**	**Recall**
Wiese et al.	all	0.693	0.420	0.764	0.920
	$contactlist$	0.683	0.460	0.680	0.843
	$somecomm$	0.707	0.724	0.488	0.467
Our dataset	$fullbook$	0.928	0.338	0.615	0.976
	$someEn1$	0.936	0.398	0.547	0.964
	$someEn10$	0.943	0.425	0.51	0.959

C. Aggregation analysis

Evaluation settings. Now we present the accuracy values for the Mobile Communication-only, the Proximity-only and the Aggregation models. Recall from III-A that only 367 pairs of users appear both in the mobile communication network channel and the physical proximity channel. Thus, to ensure equal input data in comparison of models, we use only data belonging to pairs of users appearing in both communication channels. We balanced ground truth using SMOTE and used SMO with 5-fold cross-validation as the classifier (10-fold cross-validation was not possible due to low number of input pairs).

TABLE II: Attributes and their weights in proximity and communication models.

Model	Attribute	Weight
Communication	Total duration of call	-0.0616
	Count of days with at least 1 call	-0.3544
	Count of calls	-0.0179
	Count of calls and SMSes	-0.0741
	Call duration mean time	-1.5441
	Mean time between two calls	-0.4804
	90th percentile of time between two calls	-1.534
	75th percentile of time between two calls	0.4843
	50th percentile of time between two calls	0.0186
	25th percentile of time between two calls	-0.9565
	90th percentile of call duration	-0.0237
	75th percentile of call duration	-0.0192
	50th percentile of call duration	-0.4215
	25th percentile of call duration	-0.311
	Count of days since last call	0.0298
Proximity	Count of co-location events	-0.1276
	Count of all encounters	0.4214
	Mean encounter time	-0.082
	Count of encounter days	0.1409
	95th percentile of encounters	-0.4808
	90th percentile of encounters	-0.3974
	80th percentile of encounters	-0.5826
	75th percentile of encounters	-0.1368
	50th percentile of encounters	0.3816
	25th percentile of encounters	0.6159
	90th percentile of time between two encounters	0.3849
	75th percentile of time between two encounters	0.1433
	50th percentile of time between two calls	0.092
	Mean time between two encounters	0.1584
	Count of non-encounter co-location events	0.53
	Count of days since last co-location	0.1467
	Sum of all times between two encounters	-0.2853

Results. The accuracy for the Mobile Communication-only, the Proximity-only and the Aggregation model equal 72.49%, 62.23% and 72.71% respectively. The Aggregation model obtains a 10% accuracy improvement over the Proximity-only model. However, results achieved by the Mobile Communication-only and the Aggregation models are almost equal with a slight edge for the latter (see Table III for details). Thus, although the Aggregation model achieves the best accuracy, the significant accuracy gain anticipated by Wiese et al. [1] is not observed with this dataset.

TABLE III: Classification performance with different models

Model	**F-measure**		**Accuracy**
	Strong ties	**Weak ties**	
Mobile Communication-only	0.688	0.754	72.49%
Proximity-only	0.656	0.581	62.23%
Aggregation	0.69	0.756	72.71%

Statistical Analysis. To verify that there is enough evidence to accept our claims about results, we run the statistical test. Table IV lists the accuracies of five folds for each of the three models.

We verify our claims by testing two hypotheses:

Null Hypothesis 1 (H1): There isn't any significant difference between accuracy results achieved by the Mobile Communication-only model and the Aggregate model.

Null Hypothesis 2 (H2): There isn't any significant difference between accuracy results achieved by the Proximity-only model and the Aggregate model.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

TABLE IV: Accuracy for Each Fold

	Accuracy		
	Mobile Communication-only	**Proximity-only**	**Aggregation**
fold 1	74.74%	56.84%	74.74%
fold 2	76.6%	57.45%	75.53%
fold 3	76.59%	60.64%	76.59%
fold 4	65.96%	63.83%	65.96%
fold 5	70.21%	57.45%	71.28%

We found that difference in accuracy between the Aggregate model and the Proximity-only method is statistically significant in the 95% confidence interval ($t^*_{H2} = -5.792$). However, there is not any significant difference between the accuracy of the Mobile Communication-only model and the Aggregate model in the 95% confidence interval ($t^*_{H1} = 0$).

V. Related Work

Motivated by constant increase in the use of OSNs, the research community has worked on several solutions for the social relationships based access control. Fogues et al. [31] reviewed some *Relationship-based Access Control (ReBAC)* models and specified their features. One feature which can be used for differentiating relationships in these models is tie strength. Carminati et al. [32] defined several *access control rules* and leverage relationship types for determining numerical values for strength of friendship.

Another group of research activities concern mappings of tie strength estimations between social networks. *WeMeddle*, the Twitter application showed that a predictive model for tie strength can be generalized to other social media [21]. Tang et al. [22] described a transfer-based factor graph (TranFG) model that can be used to learn and infer tie strength across heterogeneous networks.

Another set of activities is related to prediction of online social network evolution. Wang et al. [33] discovered that online and offline movement patterns have strong correlation with each other and measured that both patterns can be used for *link prediction*. They also observed that tie strength has more correlation with offline proximity than online measures. On the other hand, Kahanda et al. [34] investigated *link strength prediction* in online social networks. They derived four categories for social features and showed that network transactional features (e.g. wall posts) are the most important one.

Another field of research studies mechanisms for*trust inference* based on tie strength estimation. In [35], Seyedi et al. introduced a *proximity-based* method for bootstrapping trust values, and showed by experiment that trust values are relevant to tie strength using the MIT Reality Mining dataset. TidalTrust [36], SUNNY [37], H-OSTP [38], SocialTrust [39], FuzzyTrust [40] algorithms are examples for inferring trust in social networks.

Onnela et al. [11] examined social communication patterns based on phone calls and SMSes. They applied duration of calls for tie strength estimation to prove existence of a relationship between tie strength and local social network structure.

VI. Conclusion

In this paper, we evaluated the three new tie strength estimation models. Two of them are based on a single communication channel (the Mobile Communication-only and the Proximity-only models), while the third one (the Aggregate model) is constructed by merging all the features provided by the first two models. We evaluated performance of these models using the MIT Social Evolution dataset. Our results show a significant accuracy improvement of the Aggregate model in comparison to the Proximity-only model. However, the gain between the Aggregate model and the Mobile Communication-only model is negligible.

Based on obtained results, we cannot confirm (with this dataset) the hypothesis stated by Wiese et al. [1] that usage of multiple communication channels improves accuracy of tie strength estimation. However, their hypothesis cannot be dismissed either, as the dataset used by us contains communication data only between people that have participated in the collection campaign, thus it may not be fully representative. In addition, there are no other publicly available datasets that fulfil requirements of having multiple communication channels and verified ground truth. Finally, construction of the new dataset is also not a trivial task. In order to have a more meaningful dataset than the MIT Social Evolution dataset, it must be able to correlate identities of users (both actively participating in the dataset construction process as well as accidentally encountered) over multiple communication channels.

Acknowledgments

The authors would like to thank Mika Juuti for helping with using Weka Toolkit functions and machine learning techniques.

References

[1] Jason Wiese, Jun-Ki Min, Jason I Hong, and John Zimmerman, "You never call, you never write: Call and sms logs do not always indicate tie strength," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 765–774.

[2] Nils Jeners, Petru Nicolaescu, and Wolfgang Prinz, "Analyzing tie-strength across different media," in *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*. Springer, 2012, pp. 554–563.

[3] Blerim Cici, Athina Markopoulou, Enrique Frias-Martinez, and Nikolaos Laoutaris, "Assessing the potential of ride-sharing using mobile and social data: A tale of four cities," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, New York, NY, USA, 2014, UbiComp '14, pp. 201–211, ACM.

[4] Marcin Nagy, Thanh Bui, Emiliano De Cristofaro, N Asokan, Jörg Ott, and Ahmad-Reza Sadeghi, "How far removed are you? scalable privacy-preserving estimation of social path length with social pal," *arXiv preprint arXiv:1412.2433*, 2014.

[5] Tiffany Hyun-Jin Kim, Akira Yamada, Virgil Gligor, Jason Hong, and Adrian Perrig, "Relationgram: Tie-strength visualization for user-controlled online identity authentication," in *Financial Cryptography and Data Security*, pp. 69–77. Springer, 2013.

[6] Vikas Mittal, John W Huppertz, and Adwait Khare, "Customer complaining: the role of tie strength and information control," *Journal of Retailing*, vol. 84, no. 2, pp. 195–204, 2008.

ISBN 978-3-901882-84-5, IFIP Networking 2016

[7] Katrina Panovich, Rob Miller, and David Karger, "Tie strength in question & answer on social network sites," in *Proceedings of the ACM 2012 conference on computer supported cooperative work*. ACM, 2012, pp. 1057–1066.

[8] Wenbin Tang, Honglei Zhuang, and Jie Tang, "Learning to infer social ties in large networks," in *Machine Learning and Knowledge Discovery in Databases*, pp. 381–397. Springer, 2011.

[9] Valerio Arnaboldi, Andrea Guazzini, and Andrea Passarella, "Egocentric online social networks: Analysis of key features and prediction of tie strength in facebook," *Computer Communications*, vol. 36, no. 10, pp. 1130–1144, 2013.

[10] Jason J Jones, Jaime E Settle, Robert M Bond, Christopher J Fariss, Cameron Marlow, and James H Fowler, "Inferring tie strength from online directed behavior," *PloS one*, vol. 8, no. 1, pp. e52168, 2013.

[11] J-P Onnela, Jari Saramäki, Jorkki Hyvönen, György Szabó, David Lazer, Kimmo Kaski, János Kertész, and A-L Barabási, "Structure and tie strengths in mobile communication networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 18, pp. 7332–7336, 2007.

[12] Huiqi Zhang and Ram Dantu, "Predicting social ties in mobile phone networks," in *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on*. IEEE, 2010, pp. 25–30.

[13] Frank Bentley and Ying-Yu Chen, "The composition and use of modern mobile phonebooks," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 2749–2758.

[14] Mangesh Gupte and Tina Eliassi-Rad, "Measuring tie strength in implicit social networks," in *Proceedings of the 4th Annual ACM Web Science Conference*. ACM, 2012, pp. 109–118.

[15] David J Crandall, Lars Backstrom, Dan Cosley, Siddharth Suri, Daniel Huttenlocher, and Jon Kleinberg, "Inferring social ties from geographic coincidences," *Proceedings of the National Academy of Sciences*, vol. 107, no. 52, pp. 22436–22441, 2010.

[16] Igor Bilogrevic, Kévin Huguenin, Murtuza Jadliwala, Florent Lopez, Jean-Pierre Hubaux, Philip Ginzboorg, and Valtteri Niemi, "Inferring social ties in academic networks using short-range wireless communications," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM, 2013, pp. 179–188.

[17] Hua Wang, Vincent Chua, and Michael A Stefanone, "Social ties, communication channels, and personal well-being a study of the networked lives of college students in singapore," *American Behavioral Scientist*, p. 0002764215580590, 2015.

[18] Desislava Hristova, Mirco Musolesi, and Cecilia Mascolo, "Keep your friends close and your facebook friends closer: a multiplex network approach to the analysis of offline and online social ties," *arXiv preprint arXiv:1403.8034*, 2014.

[19] Anmol Madan, Manuel Cebrian, Sai Moturu, Katayoun Farrahi, et al., "Sensing the" health state" of a community," *IEEE Pervasive Computing*, , no. 4, pp. 36–45, 2012.

[20] Mark S Granovetter, "The strength of weak ties," *American journal of sociology*, pp. 1360–1380, 1973.

[21] Eric Gilbert, "Predicting tie strength in a new medium," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 2012, pp. 1047–1056.

[22] Jie Tang, Tiancheng Lou, and Jon Kleinberg, "Inferring social ties across heterogenous networks," in *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012, pp. 743–752.

[23] Eric Gilbert and Karrie Karahalios, "Predicting tie strength with social media," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 211–220.

[24] Tasos Spiliotopoulos, Diogo Pereira, and Ian Oakley, "Predicting tie strength with the facebook api," in *Proceedings of the 18th Panhellenic Conference on Informatics*. ACM, 2014, pp. 1–5.

[25] Nathan Eagle, Alex Sandy Pentland, and David Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15274–15278, 2009.

[26] Vedran Sekara and Sune Lehmann Jørgensen, "The strength of friendship ties in proximity sensor data," *PL o S One*, vol. 9, no. 7, 2014.

[27] Stanley Milgram, "The Familiar Stranger: An Aspect of Urban Anonymity," in *The Individual in a Social World*, pp. 51–53. Addison-Wesley, Reading, MA, USA, Aug. 1977.

[28] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.

[29] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, June 2002.

[30] Chih-Chung Chang and Chih-Jen Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.

[31] Ricard Fogues, Jose M Such, Agustin Espinosa, and Ana Garcia-Fornes, "Open challenges in relationship-based privacy mechanisms for social network services," *International Journal of Human-Computer Interaction*, vol. 31, no. 5, pp. 350–370, 2015.

[32] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham, "Semantic web-based social network access control," *computers & security*, vol. 30, no. 2, pp. 108–115, 2011.

[33] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi, "Human mobility, social ties, and link prediction," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1100–1108.

[34] Indika Kahanda and Jennifer Neville, "Using transactional information to predict link strength in online social networks," *ICWSM*, vol. 9, pp. 74–81, 2009.

[35] Amir Seyedi, Rachid Saadi, and Valérie Issarny, "Proximity-based trust inference for mobile social networking," in *Trust Management V*, pp. 253–264. Springer, 2011.

[36] Jennifer Ann Golbeck, "Computing and applying trust in web-based social networks," 2005.

[37] Ugur Kuter and Jennifer Golbeck, "Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models," in *AAAI*, 2007, vol. 7, pp. 1377–1382.

[38] Guanfeng Liu, Yan Wang, Mehmet A Orgun, and Ee-Peng Lim, "A heuristic algorithm for trust-oriented service provider selection in complex social networks," in *Services Computing (SCC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 130–137.

[39] James Caverlee, Ling Liu, and Steve Webb, "Towards robust trust establishment in web-based social networks with socialtrust," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 1163–1164.

[40] Mohsen Lesani and Saeed Bagheri, "Fuzzy trust inference in trust graphs and its application in semantic web social networks," in *Automation Congress, 2006. WAC'06. World*. IEEE, 2006, pp. 1–6.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Group-based Communication in WhatsApp

Michael Seufert*,Tobias Hoßfeld†, Anika Schwind*, Valentin Burger*, Phuoc Tran-Gia*
*Institute of Computer Science, University of Würzburg, Würzburg, Germany
{seufert | anika.schwind | valentin.burger | trangia}@informatik.uni-wuerzburg.de
†Chair of Modeling of Adaptive Systems, University of Duisburg-Essen, Essen, Germany
tobias.hossfeld@uni-due.de

***Abstract*—WhatsApp is a very popular mobile messaging application, which dominates todays mobile communication. Especially the feature of group chats contributes to its success and changes the way people communicate. The group-based communication paradigm is investigated in this work, particularly focusing on the usage of WhatsApp, communication in group chats, and implications on mobile network traffic.**

***Index Terms*—Group-based communication, WhatsApp, mobile messaging application**

I. Introduction

As the Internet has become omnipresent, nowadays a complex interplay of Internet technology and human behavior can be observed. On the one hand, the Internet is changing our daily lives, the society as well as industry and business. On the other hand, the Internet technology is driven by the adoption of the end users and stakeholders in the ecosystem.

In particular, we have a closer look at the de facto communication applications in the Internet. We notice that the applications changed the usage behavior of users. YouTube, being a prominent example of a video streaming service, made it possible to upload and stream user-generated videos, which lead to an unprecedented increase of global Internet traffic. Another change of user behavior can be seen with mobile messaging applications recently. With these applications, such as WhatsApp, users are now communicating asynchronously in groups, which are created spontaneously or which exist over a longer period. The users activity is triggered by events in these groups, i.e., posted messages, which can be enriched by user-generated images and videos. Thus, as users are always online and interacting due to smartphones and network connectivity everywhere, their activity patterns are changing.

Then again, the emerging user behavior may also change the underlying Internet technology. This happened in the past and may also happen in the future. Coming back to the same examples mentioned above, we observed the need for content delivery networks (CDNs) due to the increased video demand on YouTube. CDNs allowed to place and cache the content closer to the users and to take into account regional or social interests. This new Internet technology subsequently spread, and now many different types of applications rely on CDNs. Similarly, the changing usage behavior of group-based communication in WhatsApp might have possible disruptive implications for the future Internet.

ISBN 978-3-901882-83-8 © 2016 IFIP

The mobile messaging applications establish a publish-subscribe paradigm on application layer, which may be efficiently implemented on the network layer. Moreover, the exchange of user-generated content in groups fosters caching approaches close to the edge and the social groups. Thus, research proposals like information-centric networking (ICN) could introduce benefits. However, the increasing privacy awareness of users might lead to encrypted data communication hindering network management for ISPs. It might not yet be obvious, which technology will be employed to cope with the new challenges and demands. Nevertheless, it is the user behavior that dictates the path of technology through service acceptance and adoption.

In this paper, we show researchers how group-based communication changes the activity patterns of multiple users. This should be taken into account, e.g., when evaluating communication technologies. Models from the past (e.g., Poisson arrival process of end-to-end voice calls) cannot be directly used for nowadays applications (e.g., WhatsApp messages are exchanged in groups of users) and need to be adapted to integrate interaction of users on a smaller time scale. Therefore, we present measurement results of WhatsApp and the group communication behavior and discuss possible implications of this emerging communication paradigm on networking technology.

Section II describes the evolution of communication paradigms from one-to-one communication towards group-based communication. Section III introduces WhatsApp, the most popular application for group communication. The implications of the usage of WhatsApp on user activity and the network traffic are discussed in Section IV, and Section V concludes.

II. Evolution of Communication Paradigms

During the last two decades, the evolution of technology and especially of the Internet changed our message telecommunication. Figure 1 shows this evolution by presenting some important messaging services in chronological order of their release. It furthermore assigns them to three different ways of communication: one-to-one, one-to-many and group communication.

The 1990s were characterized by private, immediate exchange of messages between two equal partners of communications. This way of communication is called one-to-one communication. In the middle of the 1990s, for the first time

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

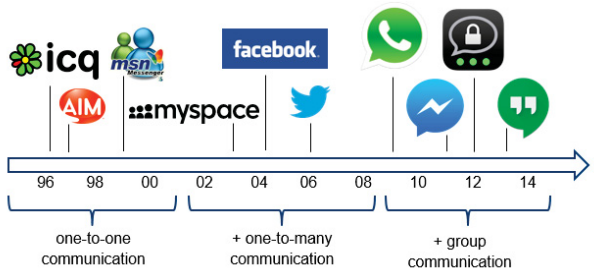


Fig. 1. Evolution of communication paradigms and notable messaging services

the concept of personal instant messaging over the Internet became available in the form of so called instant messengers. Instant messengers are chat programs, which allow users to instantly exchange messages of short size with each other. In that, they are similar to the Short Message Service (SMS), which already became available to mobile phone users in the early 1990s.

With the release of the instant messenger ICQ[1] in 1996, an alternative to the traditional email communication became popular on personal computers. In comparison to emails, which are mostly written formal, containing long and fully formulated sentences and usually have a longer response time, instant messengers are made to communicate with others similar to a real life oral conversation. Here, the content is informal with mainly short sentences. Also, a quick response is expected, so that a fluent conversation results [1]. Other well-known instant messengers are, for example, Microsoft's discontinued MSN Messenger and AIM[2], the instant messenger of AOL.

After the turn of the millennium, a new trend in online communication emerged. For the next couple of years, one-to-many communication became more and more popular. One-to-many communication is a way of broadcasting messages to a receiving group of users. Messages are simply published and can be read by everyone or by a restricted set of users, which were given rights to read the contents, or which subscribed to the feed. The messages are not necessarily answered by the recipients directly, as the main idea is to make messages available to many people at the same time. Nevertheless, it is possible to reply to messages by broadcasting an answer, so the communication is not unidirectional. This way of communication is mainly used in online social networks. These networks also support one-to-one communication, but the main usage is one-to-many communication.

In 2003 and 2004, two of the most popular online social networks, Myspace[3] and Facebook[4], were launched. This development increased the desire of users to publish messages within their social environment. A prime example for this kind of communication is also Twitter[5], which is an online service started in 2006. Twitter allows users to broadcast short messages with up to 140 characters to their so called followers, which are passive recipients.

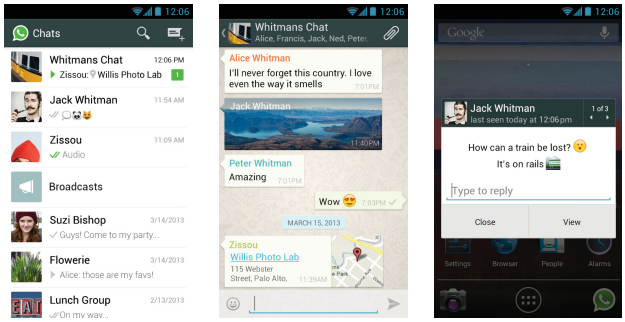
Since the middle of the 2000s, online communication has been supplemented by group communication. In this context, group communication means a conversation of a fixed group of users, which can equally participate. This process started in 2007, when the first iPhone was introduced and changed mobile communication significantly. Since this time, smartphones have become more and more popular, whereas SMS has been pushed into the background by the increasing usage of Mobile Messaging Applications (MMA). These applications are a form of instant messengers for smartphones. In contrast to emails and online social networks, MMAs are designed to allow immediate responses in real time similar to instant messengers. Additionally, communication in MMAs is not limited to one-to-one or one-to-many conversations, as many MMAs provide group conversation features. Older technologies like mailing lists and IRC also provided the ability to communicate in groups, but only with the introduction of MMAs this type of communication became widely popular. A further advantage of MMAs is the mobility, which allows to easily communicate with others from anywhere and at any time. One of the most popular MMAs is WhatsApp[6], which will be covered in more detail later [2].

Online social networks like Facebook also follow the trend of mobile communication and offer their own MMAs like the Facebook Messenger[7]. Moreover, they also added group communication explicitly, e.g., Facebook, or implicitly, e.g., Twitter hashtags, which are implicitly forming theme-based groups. Thus, the conversation in groups, the possibility of immediate responses, and the omnipresence of smartphones move written online communication further into the direction of real-life conversations.

In the future, data security and privacy, integration into the cloud, and the availability of the same services on all devices will be main points of interest. First steps in this direction have been taken, for instance, by Telegram[8], an MMA founded in 2013. This application offers end-to-end encryption of chats and cloud features. A further trend is unified communication, especially in the professional area. Unified communication combines real-time communication services, such as instant messaging or IP telephony, with non-real-time communication services, like email or fax. Finally, the support of the group communication paradigm by communication networking technology is expected. This means that traffic management solutions take into account group communications, especially in mobile networks, for example, by implementing the publish-subscribe pattern.

[1]ICQ. http://www.icq.com/ – Accessed: February 15, 2016
[2]AIM. http://www.aim.com/ – Accessed: February 15, 2016
[3]Myspace. http://www.myspace.com/ – Accessed: February 15, 2016
[4]Facebook. http://www.facebook.com/ – Accessed: February 15, 2016
[5]Twitter. http://www.twitter.com/ – Accessed: February 15, 2016
[6]WhatsApp. http://www.whatsapp.com/ – Accessed: February 15, 2016
[7]Messenger. http://www.messenger.com/ – Accessed: February 15, 2016
[8]Telegram. http://www.telegram.org/ – Accessed: February 15, 2016

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP



(a) Chat overview. (b) Conversation in a group. (c) Pop-up notification.

Fig. 2. Screenshots of WhatsApp, retrieved from the official website of WhatsApp.

III. Background on WhatsApp

WhatsApp Inc. was founded by Jan Koum and Brian Acton in Santa Clara, California, in 2009. Starting as an iPhone application, WhatsApp soon became more popular and also available for Android, Windows Phone, BlackBerry, and Nokia. In February 2014, Facebook Inc. bought WhatsApp for USD 19 billion. In September 2015, it had more than 900 million monthly active users worldwide [3], being especially popular among young people [4] and reaching a usage share of up to 77% of mobile Internet users in some countries [5].

It is very simple to start using WhatsApp because it is free to download and no complex registration is required. The application automatically identifies users by their phone numbers. Those contacts saved on the smartphone that are also users of WhatsApp will be automatically added to the application's contact list.

WhatsApp combines one-to-one, one-to-many, and group communication by offering private chats, broadcasts, and group chats. In the beginning of 2015, a desktop client and a feature for voice calls via VoIP was added [6].

Figure 2 shows various features of WhatsApp (in this example on an Android device). In Figure 2a, there is an overview over all chats and broadcasts. On top of this list is a toolbar. A private chat can, for example, be started by touching the message icon on the toolbar and choosing a contact of your list.

To start a new group, one has to open the menu and choose new group. Then, the subject of the group (a free text) must be defined and a group icon can be uploaded. Afterwards, up to 100 contacts from the contact list can be invited to join the group. The creator of a group has administrative privileges and can add and remove people from the group at any time and also promote other group members to group administrators.

An example for a group chat can be seen in Figure 2b. Each posts of a member of the group is represented by a speech bubble. Apart from the exchange of text messages, WhatsApp also allows to send photos, videos, and audio files, contact data, as well as the current location of the user. In a conversation, every type of message is seamlessly integrated into a single view, as Figure 2b shows. Every user of WhatsApp will be notified as soon as a new post arrives, whether in a group or in a private chat. This notification can be a sound, an icon, or a pop-up window, which is depicted in Figure 2c.

In contrast to SMS, WhatsApp needs an Internet connection to send and receive messages. For this purpose, it uses the Extensible Messaging and Presence Protocol (XMPP) [9]. WhatsApp is a fully centralized service, i.e., it is a service, which is operated exclusively by the US based cloud provider SoftLayer [7] This work will investigate the way users communicate using WhatsApp. Particularly, the focus will lie on group-based communication.

IV. Implications of Group-based Communication

To analyze the usage of WhatsApp and the implications of group-based communications, we conducted a measurement study on group communications, as well as a survey on the campus of the University of Würzburg, Germany in November 2014 [8]. The survey was divided into three different parts: demographic questions, group communications, and network usage statistics. The participants answered the questions of the survey in a dedicated room using personal or laptop computers, which took around 15 minutes. Questions had to be answered using text fields, single choice, or multiple choice options.

In total, 243 persons participated in the survey, which all had WhatsApp installed on their smartphones. After filtering out invalid or inconsistent answers, 209 participants remained – 106 female and 103 male. The ages of the participants ranged from 17 to 29. The average age was 21.4, which is because mostly students took part in the study. After taking part in the survey, the participants were asked to send us some of their messaging histories from WhatsApp groups by email. In that way, 402 messaging histories have been collected.

A. Usage of WhatsApp

Comparing the usage of WhatsApp to SMS, the survey showed that WhatsApp is used significantly more often than SMS. 85.17% of the participants use WhatsApp at least once or twice a day, whereas only 6.69% use SMS so frequently. This leads to the conclusion that WhatsApp communication was preferred considerably to SMS communication by the participants. The participants also had to indicate if and which other mobile messaging applications they use besides WhatsApp. Most participants (81.82%) also use other MMAs showing that WhatsApp is not the only well-established MMA. The reason is that competitive mobile messaging applications (e.g., Facebook Messenger, Skype, Threema) provide additional or different features than WhatsApp.

The participants were also asked for which purposes they used WhatsApp. 98.09% of the participants answered that they used WhatsApp for private purposes, 92.34% for organizational purposes, 77.51% for fun, 50.24% for important issues, and 33.01% for professional purposes. It follows that,

[9]The XMPP Standards Foundation. http://xmpp.org/ – Accessed: February 15, 2016

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

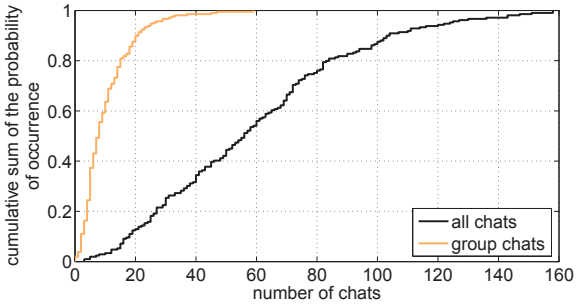


Fig. 3. CDF of the number of all chats compared to the number of group chats

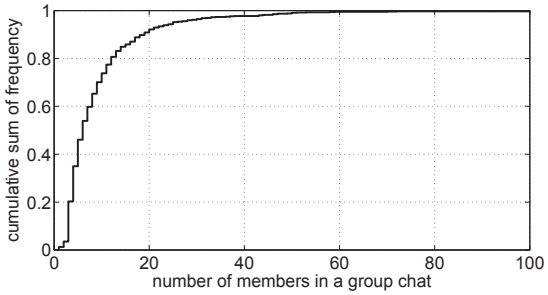
for many people, WhatsApp has become an important means of communication in many conditions of life. Moreover, WhatsApp and group communication seems to be so useful for organizing things that it also qualifies for important and professional purposes.

B. Group Chatting in WhatsApp

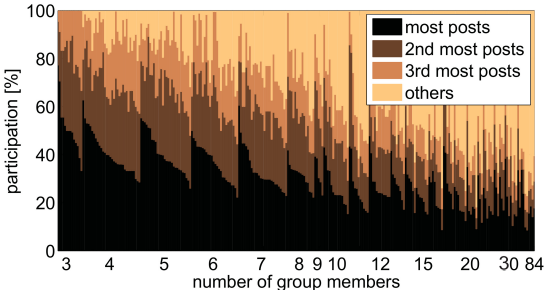
The high popularity of WhatsApp can be seen in the number of different communication chats per user. A chat in WhatsApp can be either a one-to-one chat with only one communication partner or a group chat with group sizes of 3 to 100. Thus, the participants had to count how many chats in total they had on their devices and how many of these chats were group chats.

Figure 3 depicts these distributions by comparing the number of all chats (including group chats) to the number of group chats. The average number of chats is 59. 70.32% of the participants exchanged messages with at least 30 different partners. 12.44% of the participants had even more than 100 chats, the maximum being 158. As can be seen in the brown curve, the number of groups ranged up to 59, the average being 10. Only 1.91% of the participants did not participate in any group chat. Please note that it is possible to delete group chats and the participants estimated during the study that they had already deleted on average 7 groups. Still the share of group chats among all chats is fairly high having an average of 17.94%. For most of the participants (83.28%), this share ranged between 5% and 30%. All in all, this supports the assumption that the group chat feature is used frequently by almost every WhatsApp user, which makes it a key feature of WhatsApp.

We further asked about each group chats and the participants had to specify the number of members of each group, the number of members they did not know, and their personal participation in the group chat. Figure 4a shows the distribution of group chat sizes. The average number of group members is 9, and on average one of them is unknown to the user, i.e., not in his contact list. The average group size is considerable, but low considering that WhatsApp allows creating group chats with a maximum of 100 members. Only few group members are unknown, which leads to the assumption that group chats are mainly used for communicating with specifically selected



(a) Distribution of group chat members.



(b) Active participation of group chat members.

Fig. 4. Distribution of the number of members and their active participation.

people who know each other. Nevertheless, we see that WhatsApp, like online social networks, is able to link people who do not yet know each other (triadic closure).

Next, each collected messaging history was analyzed with respect to how much posts each member actually sent in relation to all sent posts in a group chat, which is visualized in Figure 4b. It shows a bar for each group that indicates the shares of the top-3-contributors and the share of the remaining members. The bars are sorted by group size and the share of the user who sent the most posts. It can be seen that group conversations are rarely balanced. In 8.1% of the group chats, there was one user who dominated the group and sent more than 50% of all posts in this group chat. Furthermore, in 19.2% of the group chats, there were two or three members who dominated, each sending 30% or more of all posts. Also for many large groups, it can be observed that the top-3-contributors account for a quite big share of messages. This leads to the conclusion that most group chats consist of few active, dominating and several passive members. The active members send most posts in a group chat while the others in most cases only read the messages.

C. Impact on Network Traffic

Investigating the impact of group communication on the network, it is important to understand how often messages have to be transmitted. The black line in Figure 5 shows the distribution of the inter-arrival times of posts in group chats. Here the x-axis represents the inter-arrival time in minutes and the y-axis the cumulative distribution. It can be seen that many posts are replied very fast, but some messages have a very

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

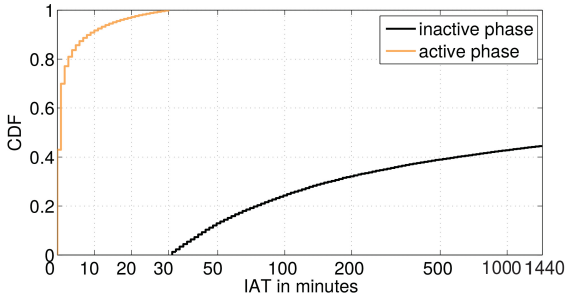


Fig. 5. Inter-arrival times (IAT) of messages in active and inactive phases

large inter-arrival time. 59.60% of the posts were immediate responses and had an inter-arrival time of one minute or less. Considering the range from 0 to 30 minutes, it can be seen that 84.90% of all inter-arrival times are included. Only 15.10% of all inter-arrival times are 30 minutes or longer. This suggests that there are different communication phases in group communications, which resembles older on-off-models (e.g., Markov-modulated Poisson processes) in telephony or networking [9].

Thus, in Figure 5, we also separate the group communication into active and inactive phases. An inter-arrival time equal or lower than 30 minutes, which is the default session timeout for web servers such as Apache Tomcat, is assigned to an active phase, higher than 30 minutes to an inactive phase. The active phase is plotted in orange on a linear scale from 0 to 30 minutes, and the inactive phase is shown in brown on a logarithmic x-axis from 30 minutes to almost two years (~106 minutes). More than two thirds (69.92%) of the inter-arrival times in active phases were immediate answers in one minute or less. In inactive phases, 44.42% of the inter-arrival times are one day or less and almost all messages are replied within one week (~104 minutes). The probability of changing from an active to an inactive phase is 12.24%, from inactive to active it is 69.99%. This distributions support the statement that WhatsApp constitutes a very fast communication.

Over the course of a day, the typical diurnal pattern can be observed with most posts being sent in the evening from 6pm to 8pm (15.10%), fewest between 5am and 7am (0.44%). This also confirms the statement of the participants that only few use WhatsApp for professional purposes. The participants were asked to enter the statistics of WhatsApp's network usage, which are collected by the application on every device. Generally, the communication of each user in WhatsApp is balanced, however, an average user receives roughly 21% more messages than he actually sends. The slightly higher rate of received messages is likely to be caused by group chats. Recall that in these chats, every message sent by one user is potentially received by a multitude of other users. We observed in the messaging histories that media posts, e.g., photos, videos, or voice messages, are sent very rarely. On average only 6.53% of all posts in a group were media posts. However, considering the relation between received media

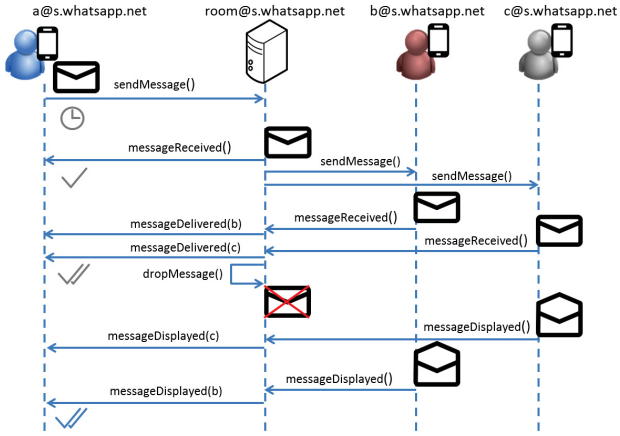


Fig. 6. Process of sending a message in a group chat with three users

bytes and received bytes in total, nearly 86% of the total bytes can be attributed to media posts. This ratio has a linear behavior among our participants with high Pearson correlation coefficient of 0.92. This indicates that media posts generally cause the largest part of WhatsApp's network traffic. Similar findings obtained from passive measurements in a cellular network are presented in [7]. With respect to message sizes, a simple measurement showed that, as expected, the size of a text message increases linearly with the number of characters. During our study, we observed an average message length of 36 characters, which resulted in an average text message size of 317B. For media messages, [7] reports an average size of 225KB. However, a more thorough investigation is needed, as we noticed during our measurement that WhatsApp applies transcoding and scaling to transmitted images and videos.

Not only the data volumes of the messages themselves, but also a lot of application-layer signaling traffic puts load on the mobile network. Figure 6 illustrates the process of sending a message in a minimal group chat with only three users following the specifications of the XMPP protocol [10]. User A sends a message to the server addressing it to the room of the group. The message can be a text post or media post and its content can potentially have a high data volume. As soon as the server receives the message, it sends an acknowledgement to user A. With a simple setup, we measured that sizes of acknowledgements are around 60B. Upon arrival of the acknowledgement, the clock symbol turns into a single grey checkmark on user As phone. The server forwards the message with the content to all members of the group, which are user B and user C. The message receptions are acknowledged to the server. The server forwards the acknowledgements to user A, signaling that the message was delivered. As soon as the message is received and acknowledged by all members of the group, the server drops the message and the symbol in the chat turns into a grey double checkmark. If a user in the group displays the message to eventually read it, it is reported to the server, which forwards it to the sender.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

After all group members were reported to have displayed the message, the double checkmark turns blue. Hence, sending a message within a group implies a number of subsequent messages signaling the reception and processing state of the message of each user. Additionally, each application signaling induces a lot of signaling in the mobile network [11]. In groups with many members this results in a significant traffic volume and high number of signaling messages, which have to be processed by the network. In peak hours or in case of flash crowd events, this may lead to problems in the network and requires management of the traffic.

V. Conclusion

In this paper, we investigated group-based communication in WhatsApp. Communication in groups constitutes an emerging communication paradigm, which has a huge impact on todays mobile networks. We conducted a survey on WhatsApp usage and analyzed collected messaging histories to better understand group communication and its impact on network traffic.

All in all, this work provided a first investigation of group-based communication in WhatsApp, which changes the way how people communicate. The analyses presented in this paper allow for modelling and simulating communication in groups. Thereby, novel traffic management mechanisms can be designed and evaluated in order to better cope with the network demands. These might include the ICN proposal and caching of content close to the end users, publish-subscribe mechanisms, or multicast transmissions instead of transmitting content naively to each individual group member. Moreover, other approaches like mobile ad hoc transmissions could become relevant, especially in cases in which (parts of) the virtual WhatsApp group physically meet. Content could then be exchanged directly when the members are in the same WiFi network or by short-range device-to-device communication as currently discussed for 5G networks. It remains for future work to analyze the benefits and to study the applicability of each approach in order to adapt the current Internet technologies to the changing user behavior with group communication being well underway.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grants HO4770/1-2 and TR257/31-2 (DFG project OekoNet). The authors alone are responsible for the content.

References

[1] A. F. Cameron and J. Webster, "Unintended Consequences of Emerging Communication Technologies: Instant Messaging in the Workplace," *Computers in Human Behavior*, pp. 85–103, 2005.

[2] C. Montag, K. Błaszkiewicz, R. Sariyska, B. Lachmann, I. Andone, B. Trendafilov, M. Eibes, and A. Markowetz, "Smartphone Usage in the 21st Century: Who is Active on WhatsApp?" BMC Research Notes 8.1: 331, Tech. Rep., 2015.

[3] L. Rao, "WhatsApp hits 900 million users," 2015. [Online]. Available: http://fortune.com/2015/09/04/whatsapp-900-million-users/

[4] J. Fetto, "The $19 billion question: Who uses WhatsApp and why are they so important to Facebook?" 2014. [Online]. Available: http://www.experian.com/blogs/marketing-forward/2014/02/21/the-19-billion-question-who-uses-whatsapp-and-why-are-they-so-important-to-facebook/

[5] J. Mander, "WhatsApp Usage Highest in LatAm and MENA," 2014. [Online]. Available: https://www.globalwebindex.net/blog/whatsapp-latam-mena

[6] A. Chowdhry, "WhatsApp Android App Now Has Free Voice Calling For Everyone," 2015. [Online]. Available: http://www.forbes.com/sites/amitchowdhry/2015/03/31/whatsapp-calls-android/

[7] P. Fiadino, M. Schiavone, and P. Casas, "Vivisecting Whatsapp Through Large-scale Measurements in Mobile Networks," in *Proceedings of the ACM SIGCOMM*, Chicago, IL, USA, 2014.

[8] M. Seufert, A. Schwind, T. Hoßfeld, and P. Tran-Gia, "Analysis of Group-based Communication in WhatsApp," in *Proceedings of the 7th EAI International Conference on Mobile Networks and Management (MONAMI)*, Santander, Spain, 2015.

[9] D. L. Jagerman, B. Melamed, and W. Willinger, "Stochastic Modeling of Traffic Processes," *Frontiers in Queueing: Models and Applications in Science and Engineering*, pp. 271–320, 1997.

[10] P. Saint-Andre, "XEP-0045: Multi-User Chat," 2008. [Online]. Available: http://xmpp.org/extensions/xep-0045.html

[11] C. Schwartz, T. Hoßfeld, F. Lehrieder, and P. Tran-Gia, "Angry Apps: The Impact of Network Timer Selection on Power Consumption, Signalling Load, and Web QoE," *Journal of Computer Networks and Communications*, 2013.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Online Engagement and Well-being at Higher Education Institutes: A German Case Study

Margeret Hall
Karlsruhe Service Research Institute, KIT
Karlsruhe, Germany
hall@kit.edu

Simon Caton
National College of Ireland
Dublin, Ireland
simon.caton@ncirl.ie

***Abstract*—Society is increasingly mirrored in the digital sphere. Unknown is how well this maps back in real-life aspects like the feeling of well-being or community engagement. Our tool 'BeWell@KIT' was employed to parse 140 Facebook pages comprising the online social media presence of a large public German university. Discourse attributes are established and investigated to identify the interaction between digital discourse and real-world events. We find evidence of critical system-wide events directly impacting the expressed well-being and engagement of the community, while also displaying an uptick in the psychological concepts 'belongingness' and 'resilience.' Our study indicates that digital expressions of well-being support the healthy functioning of the community at large.**

Keywords—Online Social Networks; Facebook; Behavior Modeling; Sentiment Analysis; LIWC

I. INTRODUCTION

The dividing line between off- and online communities is increasingly blurred. Digital participation and communication has become the 'new normal' [1]. Considering fast-paced online communities there is an institutional interest in knowing if, and which, events have significant effects on the way the community interacts and expresses well-being online, what changes sentiment over time. Similar works name the mapping of digital sentiment and behaviors a 'Social Observatory' [2]. An interesting extension to existing efforts is the mapping of digital interactions of a university community considering expressed well-being. We ask, '*Which attributes of digitally expressed, institutional well-being can be extrapolated from informal online text?*'. To address this, our work evaluates the performance of a university from the perspective of the university community's subjective opinion(s) of itself online, aggregating based on community well-being and expressions of engagement. Such a system establishes a more granular and sensitive feedback system for stakeholders (i.e., administration, students, faculties) to assess and respond to university performance. In response to this a Social Observatory is employed to analyze and report socially-sourced indicators on university quality and satisfaction. The Social Observatory procured data from popularly used public Facebook pages surrounding the Karlsruhe Institute of Technology (KIT) [1], for a tool that is near to real time and sensitive to concerns of both privacy and the desire to participate online.

ISBN 978-3-901882-83-8 © 2016 IFIP

Section II reviews related work and Section III justifies the design made in the implementation choices and gives the descriptive attributes of the KIT Facebook network. Section IV reviews the macro, meso and micro attributes of communal discourse across the KIT Facebook network. Section V discusses and contextualizes the findings and addresses limitations, and Section VI concludes the chapter.

II. RELATED WORK

A. On the application of Social Media Platforms for Social Sentiment Analysis

In the 1960s computational innovations resulted in a challenge shift: The restricting parameter for social researchers was no longer the collection of data. Instead, information grew at a rate faster than researchers could analyze [3]. Developments in people's daily lives are at once more transparent, yet more difficult to understand. This is due in part to the rise of networked online social data. Social media sites in particular have quickly ascended from a novelty of the early 2000's to a fact of life, and daily necessity. Users interact online by creating profiles and providing (semi)personal information in form of text, photos and other media [4]. As social networking and media platforms are generally based on true identities or variants thereof [4], [5], they are well suited for digital community analyses.

Facebook is the largest, most active platform with its 1.308 billion daily active users, with one in every seven minutes worldwide (and for Americans, one in every five minutes) spent on Facebook.[1] In an exhaustive survey, [6] summarized and classified 412 articles written on Facebook for the period 2007-2012 leading to five supra-categories: descriptive analysis of users, motivations for using Facebook, identity presentation, the role of Facebook in social interactions, and privacy and information disclosure. Recognizable is that the usage of Facebook's API by non-Facebook staff or partners to support quantitative, unobtrusive studies is low; when the referenced studies apply quantitative methods, the method of choice tends to be based in survey methods.

Notable studies from Facebook Research look at public expressions of sentiment. [7] used status updates based in the United States to create a composite well-being index. Another

[1] http://techcrunch.com/2014/07/23/facebook-usage-time/. Last Accessed: 12 March 2015.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

series of studies by Kramer and colleagues [8], [9] reviews emotional contagion on Facebook. These studies report that emotions are indeed contagious in a network. Their findings support that short informal text like Facebook status updates can be used to measure sentiment online. To date, no studies were identified that dissect how educational institutions and their communities use, leverage, and engage computer-mediated communication with Facebook. This article addresses that research gap.

B. *Linguistic Inquiry and Word Count*

LIWC originally was not intended to be used on short informal text, but to analyze text of expressive and therapeutic writing sessions usually containing more content than the average tweet or Facebook update [10], [11]. However, its expansive psychometric dictionary offers a unique opportunity to reveal the latent emotional context of text-based data. The application of LIWC on documents returns the percentage of words across the categories social processes, affective processes, cognitive processes, perceptual processes, biological processes, work and achievement, as well as punctuation and structural details [11], [12]. Per cent based information gives the researcher a mechanism by which to see the relative worth of categories in speech. This facilitates measuring change, looking for group-based patterns, monitoring individual spikes and dips, and identifying psycholinguistic profiles.

When people share (written) information, there is not only content but also the way they create their message and the linguistic style [13]. They found that function words are well suited to build a systematic picture of this inconceivable dimension as latent indicators. They refer to pronouns, prepositions, articles, conjunctions, and auxiliary verbs and altogether can be imagined as "[…] the linguistic "glue" that hold content words together" [14]. While LIWC focuses on function words it also includes content words. The functionality is based on dictionaries that assign over 4,500 words to 70 different categories, ranging from a simple stylistic (e.g. article, prepositions) to a complex psychological level (e.g. positive emotion, cognitive words). Due to their near constant usage and grammatical weight, use of function words is nearly impossible to manipulate and thus will uncover motives, personality and psychological processes more accurately than analysis of the content [15]. Using computational tools in analyzing function words bears further advantages. Firstly, people's poor awareness of function words is not restricted to their own language. The listener doesn't focus on function word composition, and therefore is unable to rate usage. Hence, computational pattern matching can reveal findings not attainable by human judges. Secondly, less than 0.04% of an average persons' vocabulary are function words [13]. At the same time, they make up more than half of daily language. Consequently, function-word based analyses are well-situated to reveal latent individual states. All in all, the function word's importance on psychological findings justifies the application of its simpler dictionary-based approach wherever emphasis is set on personal traits.

III. Design and Approach

To address research questions several steps must first be taken. Following the methodology established in [1], [2] the raw data is first filtered based on post type, then aggregated to represent groups of the university; is run through LIWC, and finally mapped and assessed. In order to create a comparative baseline, LIWC scores of all data (posts and comments) before the start of the event and after its completion have been aggregated to a single number, weighted by total word counts. All measures in the coming analyses do not show the actual LIWC scores, but relative increases and decreases considering a time-local baseline. The app used to extract Facebook data is open source and can be found in [2].

The first assumption to be addressed it the use of Facebook as opposed to Twitter. The KIT database of Facebook activities features an average text length of 33.96, mainly German, words. Given Twitter's character restrictions and that the average German word length is estimated as 5.7 characters[2], this would exclude 33.57 characters of the average message or otherwise force unnatural brevity or improper spellings; consequently adding complexity and errors to the (text) analysis. The fraction of posts and comments in this procured dataset containing more than 160 letters (28 words on average) represents 80.1% of the corpus, reflecting 39.86% of all comments and posts being longer than Twitter's restriction. Twitter would certainly result in drastically shorter text submissions and consequently in a loss of more complicated, reflective statements. There is an additional restriction of Twitter that lends unknown biases, namely that Twitter grants between 1-10% of the data available from the first request date in a given query [16]–[18], compared to the full Timeline of the Facebook extraction. For KIT, Facebook usage outranked all other Social Media usage for both university-generated and student-generated content, which is in line with the fact that Facebook has an 82% market reach of Germany, whereas Twitter has approximately 20%.[3]

The 140 pages in the dataset represent all open pages which were (1) primarily populated by KIT members and (2) had posted 50 or more words between the timespan of 2011-2014. Finally, four granularities are investigated: post-comment splits, page group splits, administration-faculty splits, and temporal aspects. From this baseline it is possible to see what, if any, spikes and dips appear. Accordingly the next section describes the KIT Facebook community, establishing the attributes that make up the communal discourse.

IV. Macro, Meso, and Micro Granularities of BeWell@KIT

In order to gain a more granular understanding of how the KIT relates and interacts online, the baseline of discourse and latent emotive value are established focusing on the years 2011-2014; while some pages were open longer than this, all pages included in the study were open from 2011 onwards.

[2] http://www.duden.de/sprachwissen/sprachratgeber/durchschnittliche-laenge-eines-deutschen-wortes. Last Accessed: 10 March 2015.

[3] http://www.statista.com/statistics/280176/penetration-rate-of-social-media-sites-in-germany/. Last Accessed: 10 March 2015.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

A. Macro Attributes of the KIT Facebook Network

Table I displays descriptive characteristics of the dataset. Likes far outnumber posts and comments, and posts outnumber comments. That posts outnumber comments in this use case is a surprising characteristic as most official pages only permit administrators to post on the timeline; constituent participation is restricted to commenting on those posts.

TABLE I: SUM OF VALUES OF ALL PAGES IN KIT FACEBOOK NETWORK CONSIDERING POSSIBLE INTERACTIONS OF THE PAGES AND AUDIENCES

Page Likes	Status Updates	Wall Posts	Comments	Likes on Posts	Resources Posted	Resources Liked
101,772	**26,259**	**4,284**	**16,079**	**179,721**	**8,817**	**45,241**

Given that KIT is an online community, it is expected that its members communicate in similar time spans. KIT's communal discourse has a cyclic pattern that matches recurring semester cycles: The start of semester, mid-semester, exam weeks and semester holidays. The intensity of interactions also follows this pattern closely, as approximately 66% of interaction occurs inside of the semester. Figure 1 displays the average over academic years considering the timespan 2011-2014 in per cent. There it can be seen that the bulk of discussions occur inside of the semester, with the Winter Semester having slightly more chatter than the Summer Semester. This pattern is flipped for the holiday seasons, which Summer Holidays having a slight boost in activity compared to the Winter Holidays. That remains constant when comparing the exam weeks to the holidays – Winter Holidays have less Facebook interaction than the Winter Exams, and Summer Holidays have more interaction than the Summer Exam period.

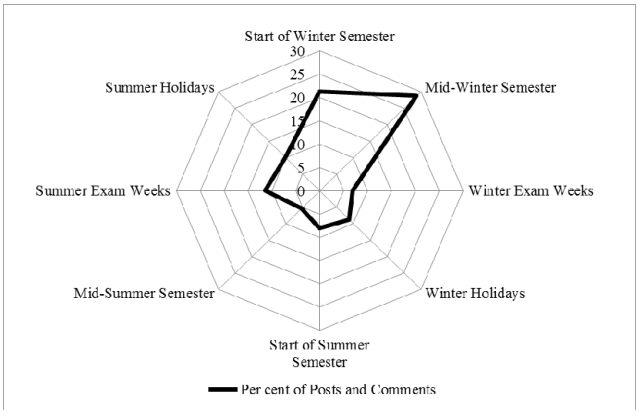


Fig 1. Frequency of posts and comments at KIT, 2011-2014

B. A Meso-assessment of KIT's Discourse Baseline

A group representation is the creation of supra-groups based on commonalities (e.g., administration, faculties, student groups) used to assess the KIT community as a more realistic replication. Regarding group partitioning, two approaches are executed. First, all the 140 available pages are assigned to one of 12 page categories in order to facilitate analyses of the university's Facebook community. The naming of the groups is guided by the KIT website where possible to assure a realistic assessment in reconstructing discourse. In the case of KIT affiliated but not KIT sponsored groups, the most general common name is used. The names of the groups are *KIT (official presence), Library, Schools, Departments and Institutes, Student Clubs, University Clubs, Sports Teams, Innovation and Development, Politics, Career, Music, and Social*. It must be noted that during the course of the study five pages closed and were duly excluded from the analysis; pages with less than 50 words over the four years of assessment are likewise excluded. These groups are further assessed considering if they are run by administrators or students. Splitting the data into these subgroups aims to reproduce an accurate picture of the community, by taking interactions and communal diversities within into account.

Comparing administrators and students reveals interesting differences in the discourse baseline. Results of an Independent Sample Mann-Whitney U test show highly significant differences in the use of Positive Emotion (U = 6,740, z = -4.520, p = .0005) and Negative Emotion (U = 7,530, z = -3.381, p = .0005), using an asymptotic sampling distribution for U. Mann-Whitney U is the non-parametric estimation of a One-Way ANOVA. Administrators show a lower frequency of positive and negative emotional discourse. When these emotions are employed, they tend to be employed by students. Net Affect, a composite variable, is calculated by subtracting negative sentiment categories from positive sentiment. KIT's network is mesokurtic with a positive skew (Figure 2a) and a reversed sigmoid distribution (Figure 2b), hovering at zero but with a long positive tail.

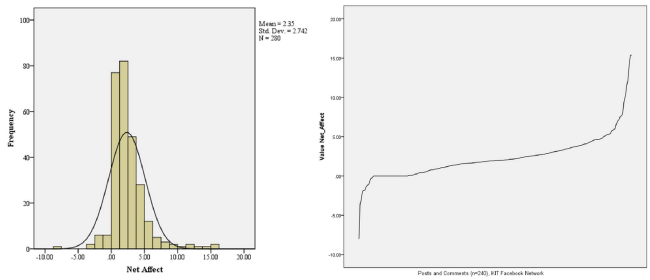


Fig. 2 Net Affect, displaying skewedness and (a) Kurtosis and (b) Distribution

That KIT's Net Affect tends to hover around zero signifies few pages employing extreme emotion. The absolute range is -8.0 from the OSKar- Optics Students Karlsruhe e.V comments to a positive 15.38 from the comments of the Institute of Regional Science. Students tend to make up both ends of the tails, and administrators are grouped in the middle of the distribution (the zero range). This supports the results of the Mann-Whitney U tests that students are move emotive than administrators in their digital discourse.

When this is considered alongside with the tendency of comments to use more cognitively expressive (Figure 3) and emotive discourse in their responses, it can be understood that although this tendency should be expected in most communities, the size of this gap indicates that the university's constituents visit the pages to seek and engage in lively

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

discussions. Comments display significantly higher cognitive complexity than posts (U = 5,831.5, z = -5.861, p = .005).

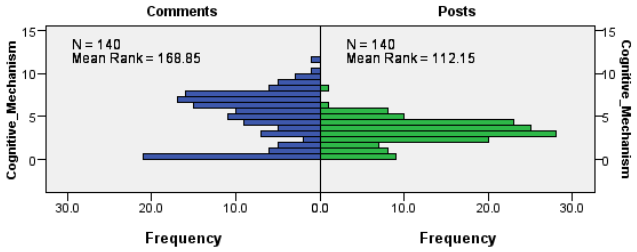


Fig. 3. Results of a Mann-Whitney U test comparing cognitive complexity

C. *Micro Representations of Communal Well-being*

Given KIT's cycle discourse, a reasonable way to identify events of impact is to inspect sentiment spikes and dips as they are related to the semester intervals. The following analysis addresses the benchmarks of the semester. BeWell@KIT established a critical system shock for students and employees as the denial of the Elite Status on 15 June 2012.[4] The loss acted as a shockwave across the network and was the most common discussion topic the days after the loss, as it was expected to damage the university's prestige and signaled the end of the 'Excellence Money,' a governmental support of 15-20€ million yearly. Students feared decreasing employment opportunities in the highly competitive academic working environment. At the same time, financial consequences threatened the continuing of research projects and existence of administration jobs. The denial impacted students, researchers, and administration employees likewise [1].

The Facebook community's overall activity after publication of the judges' Excellence decision increased strongly. Whilst the week before the announcement counts 7,425 words, this amount increases by one third to 11,070 words during the consecutive week and 15,072 (almost an additional 25%) two weeks after the event. The two weeks representing the event and after the event comprise 1.3% of the corpus' words. The categories reflecting cognitive complexity (Articles, Exclusion, Causation) show a positive trend in the following week of the Excellence loss compared to the overall score before (Table III). Putting this together with the significantly higher scores of Past and Future (measuring verb tense frequency), and the topic categories Money, Occupation, Job and School is an indication of intense discussion on the reasons and future impacts of the Elite denial.

TABLE III SCORE DEVELOPMENT FOR COMPARISON BETWEEN 1) ALL DATA BEFORE JUNE 15TH 2012, 2) THE FOLLOWING FIRST WEEK AFTER THE EVENT AND 3) THE FOLLOWING THREE WEEKS AFTER THE EVENT WHERE GREEN SHOWS INCREASES AND RED SHOWS DECREASES

	Before Loss	1 Week After	3 Weeks After
Articles	6.68	8.24	7.64
Exclusion	0.86	1.04	1.04
Causation	0.63	0.88	0.72
Past	1.31	1.85	1.71
Future	0.56	0.78	0.71
Money	0.72	0.89	0.68
Occupation	5.49	6.07	5.83
Job	1.89	2.06	2.04
School	2.87	3.37	3.19

[4] http://www.kit.edu/kit/english/5963.php. Last Accessed: 3 January 2015.

The first week shows the most distinct peaks for all cases. Still, a wider timeframe post-event produces the same tendencies for all LIWC categories but Future (Figure 4). The additional three-week timeslot enables observation whether detected peaks presume or ebb away quickly. All categories except Future display significant percentage increased at the $p < 0.05$ level in the short-term and that Occupation and Past maintain mid-term significance at $p < 0.05$.

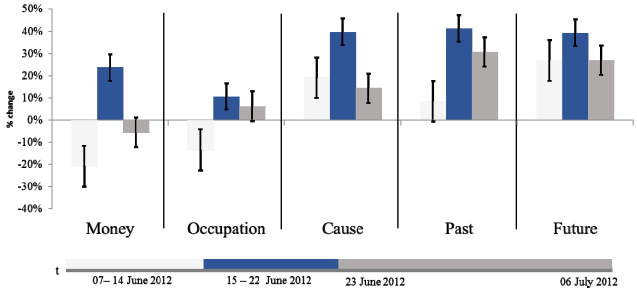


Fig. 4 Affective changes in discourse relating to the KIT Elite loss. All measures show relative changes, not absolute LIWC scores. The blue bars in the middle reflect the event week, while bars to the left (1 week before) and right (3 weeks after) represent temporal deviations from the baseline.

More than impacting professional and practical concerns, the loss of the Excellence status had a major influence on the KIT's digital expressions of well-being (Figure 5). Increased frequencies of the categories Negative Emotion and Sad hint at collective frustration. Positive Feeling depicts a decrease (-35.7%) directly after the announcement. More interesting are the categories Social and Inclusion, which are proxies for the feeling of belonging to the community of reference. The LIWC categories Social and Inclusion increase slightly after the incident, and quickly increases in the following three weeks. Here we see no significant change between the week preceding and the week of announcement. However, significant increases ($p < 0.05$) in social and inclusive discourse are seen in the month after. This indicates both expressions of increased community belongingness, and its related construct resilience. It is interesting to observe that after the event, zooming out to the following three weeks the categories show a slight upswing indicating communal resilience while reminding us how delicate results based on latent emotional states are.

These two categories are strong reflectors of communal belongingness, thus leading to an interesting finding. Because the loss was unexpected it affected almost all community members: the shock was wide-spread and deep. Former research found that tragic collective experiences often promote feelings of belongingness [7], [19], [20]. This is evident in the KIT dataset, where the loss of the Excellence status acted as a collective crisis. Encouragingly, the community responded with not only shock and negative feelings, but also resilience and an increase in togetherness, signs of well-being according to the definition of [21].

The Excellence initiative reaction suggests that campus-wide incidents affect the way the community interacts. Well-being is affected in the short-run, but the long-term impacts are

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

minimal. Belongingness increases in spite of the negative feeling overall. This highlights both community resilience, and how delicate the results are.

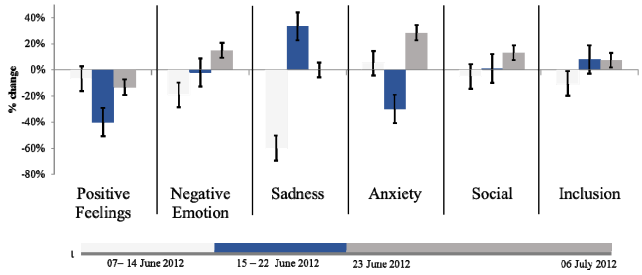


Fig. 5 Emotive sentiment flow in discourse relating to the KIT Elite loss.

V. Discussion

Focusing a Social Observatory on the KIT Facebook network revealed quite clear online discourse patterns among university network members. Post-comment comparison, in which posts represent activities of page administrators and comments participation of page visitors, serve as the analysis's baseline, providing insights into the community characteristics as a whole, and as a guideline for further data partitions.

LIWC results display an overall satisfied community, disclosing indicators of high emotional and mental well-being through various emotional, attentional and cognitive categories. Interestingly, comments are both the most positive and negative aspects of the dataset, indicating that the community has a diversity of emotion even though the net effect is overall positive. To better understand the dynamics of discourse, focus was shifted to differences between comments and posts, considering if it originated on an administrator or student led page. Comments on student pages are more emotional overall. Combining this with the prevalence of cognitive processes in comments, it can be posited that a central motivation for visiting the KIT Facebook pages is seeking lively discussions and discussion of opinions. In contrast, university administrators seem to restrict themselves to 'newsflashes' in a professional, formal manner, avoiding narratives.

With respect to the temporal aspects of the analysis, several interesting patterns were detected. Campus discourse showed dependencies with the recurring semester cycles. KIT's Facebook community is most active when students are returning from holidays to the new semester. Supplementary pressure and study habits seem to reduce social activity in contrast to the middle of the semester, where social processes peak. The denial of the Elite status acted as a shockwave not only on the campus but also across the various pages of the university's Facebook community. Members reacted emotionally with anger, anxiety and sadness summarized by a generally increased density of negative emotion. Positive feelings in the community were marked by a significant drop from the week preceding the announcement. However, the community showed resilience as displayed by an increase in positive emotions, and social and inclusive discourse three weeks after the event. Remarkably, the KIT community responded with an increase of communal belongingness to this disappointing experience.

A. Limitations and Future Work

This work focuses on spikes and dips with clear data signals in its current iteration. Innumerable smaller and unstudied incidents can add up and be responsible for emotive shifts just as well as significant and sudden dips and spikes. This is due to the fast-changing features of and in social media, including strong dynamics without distinct attributes. The long-term analysis of events seems best suited for large-scale political interventions ([22]) or small, clear communities [1].

Some limitations caused by the tools available do exist. As stated, LIWC was not designed for short informal text like that found in Online Social Media, even though it copes astonishingly well. The importance of multilingualism in Online Social Media is increasingly recognized. Interlanguage comparison or even pages including a mixture of several languages could mislead interpretation of results. To allow for consideration of these inaccuracies further software versions could process an output reflecting word count percentages of contained languages.

A major limitation of this exploratory work is its reliance on estimations of emotional states. This is especially true for dictionary-based approaches that are insensitive to context and thus will misinterpret ambiguous words and certain linguistic constructs like irony or sarcasm. Although there is a high amount of agreement with established literature to indicate this study's validity, better grounding of the dictionary to context and not only latent states would allow for more definitive statements on the general health of the community.

An interesting extension would be a comparative assessment of other universities and technical universities in Germany, as well as (dis)similar global universities. This would enable the establishment of in-depth comparisons of community characteristics and participative behavior, representing a powerful information resource for education institutions worldwide. It would also establish the findings this work as confirmatory rather than exploratory.

VI. Conclusion

BeWell has shown that it can detect notable community events by tracking expressed sentiment in Facebook posts and comments. This work's contribution is the binding of a multi-dimensional well-being definition that are otherwise hidden inside a data stream. To achieve this, both benchmarks from literature and unusual sentiment-based spikes and dips were observed and reported.

The results revealed by the temporal analysis indicate that within a community, stakeholders should not be identified in a top-down way. Especially the shockwaves across the digital community after the loss of the Elite status show that the community is both self-nominated, and highly engaged, participating in the events and emotions experienced as a community. Partitioning the data in recurring semester cycles presents information on how communication focus shifts over the year. Due to the fact that people frequently debate about

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

daily activities and events the results also capture the prevailing topics of daily activities. The way a Facebook page is administrated also seems to affect a basic indicator contributing to well-being, namely the feeling of communal belongingness. This characteristic is especially valuable for institutions since it reflects if constituents can identify themselves with values and views of the organization.

Sentiment analysis on Facebook and the KIT Facebook presences revealed multiple characteristics useful to describe a community. LIWC score interpretation allowed for the identification of the community's well-being, belongingness, resilience and engagement. The description of characteristics was not restricted to capturing macro tendencies but even delivered dynamics over time, sentiment cycles, and differences between various subgroups of the respective community. Results affirm LIWC as an efficient analysis tool for tracking communal sentiment, well-being and aspects of belongingness. The results are quite often nuanced: small percentage points highlight differences for more than one community characteristic. Yet, topic domains and specific other scores allow for detecting more specific interpretations and should not be disregarded.

Information estimated from aggregated social media data may lack some interpretation quality but provides an easy and repeatable way to gain quick insight into the essential factors defining a community. Macro-assessment of social indicators rises from investigation of post-comment distinction, a pre-given structure of any Facebook dataset. This means that the approach is easily replicable for other communities and generalizable. Although some customizing effort concerning data preparation are inevitable if community-specific insights are pursued, many of the employed partitions are to be individualized to further use cases. This aspect of popularly sourced well-being information is ripe for broader adaptation.

REFERENCES

[1] A. Lindner, C. Niemeyer, S. Caton, M. Hall, C. Niemeyer, and S. Caton, "BeWell: A Sentiment Aggregator for Proactive Community Management," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, 2015, pp. 1055–1060.

[2] S. Caton, M. Hall, and C. Weinhardt, "How do politicians use Facebook? An applied Social Observatory," *Big Data Soc.*, vol. 2, no. 2, p. 2053951715612822, Dec. 2015.

[3] C. Cioffi-Revilla, "Computational social science," *Comput. Stat.*, vol. 2, no. 3, pp. 259–271, May 2010.

[4] M. Hall and S. Caton, "A Crowdsourcing Approach to Identify Common Method Bias and Self-representation," Oxford, England, 2014.

[5] J. Lingel, M. Naaman, and danah boyd, "City, self, network: transnational migrants and online identity work," in *CSCW'14*, 2014, pp. 1502–1510.

[6] R. E. Wilson, S. D. Gosling, and L. T. Graham, "A Review of Facebook Research in the Social Sciences," *Perspect. Psychol. Sci.*, vol. 7, no. 3, pp. 203–220, May 2012.

[7] A. Kramer, "An Unobtrusive Behavioral Model of 'Gross National Happiness,'" in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 287–290.

[8] A. Kramer, "The spread of emotion via facebook," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, 2012, pp. 767–770.

[9] A. Kramer, J. E. Guillory, and J. Hancock, "Experimental evidence of massive-scale emotional contagion through social networks," *Proc. Natl. Acad. Sci.*, vol. 111, no. 24, pp. 8788–8790, Jun. 2014.

[10] N. Wang, M. Kosinski, D. Stillwell, and J. Rust, "Can Well-Being be Measured Using Facebook Status Updates ? Validation of Facebook ' s Gross National," *Soc. Indic. Res.*, vol. 115, no. 1, pp. 483–491, 2014.

[11] Y. Tausczik and J. Pennebaker, "The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods," *J. Lang. Soc. Psychol.*, vol. 29, no. 1, pp. 24–54, Dec. 2010.

[12] J. Pennebaker, C. K. Chung, M. Ireland, A. Gonzales, and R. J. Booth, "The Development and Psychometric Properties of LIWC2007," University of Texas, Austin, Austin, TX, 2007.

[13] C. Chung and J. Pennebaker, "The Psychological Functions of Function Words," *Soc. Commun.*, pp. 343–359, 2007.

[14] C. J. Groom and J. Pennebaker, "Words," *J. Res. Pers.*, vol. 36, pp. 615–621, 2002.

[15] J. Pennebaker, *The Secret Life of Pronouns: What Our Words Say About Us*. New York, New York, USA: Bloomsbury Press, 2013.

[16] S. González-Bailón, N. Wang, A. Rivero, and J. Borge-Holthoefer, "Assessing the bias in samples of large online networks," *Soc. Networks*, vol. 38, no. January, pp. 16–27, 2014.

[17] D. Ruths and J. Pfeffer, "Social media for large studies of behavior," *Science (80-.).*, vol. 346, no. 6213, pp. 1063–1064, 2014.

[18] M. Russell, *Mining the Social Web*, Second. Sebastopol, CA: O'Reily Media, 2013.

[19] J. Pennebaker, M. R. Mehl, and K. G. Niederhoffer, "Psychological aspects of natural language use: our words, our selves.," *Annu. Rev. Psychol.*, vol. 54, pp. 547–77, Jan. 2003.

[20] J. Pennebaker and T. C. Lay, "Language use and personality during crises: Analyses of Mayor Rudolph Giuliani's press conferences.," *Journal of Research in Personality*, vol. 36. pp. 271–282, 2002.

[21] F. Huppert and T. T. C. So, "What percentage of people in Europe are flourishing and what characterises them?," Florence, Italy, 2009.

[22] B. Böcking, M. Hall, and J. Schneider, "Event Prediction With Learning Algorithms—A Study of Events Surrounding the Egyptian Revolution of 2011 on the Basis of Micro Blog Data," *Policy & Internet*, vol. 7, no. 2, pp. 159–184, 2015.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Performance of real-time collaborative editors at large scale: user perspective

Quang-Vinh Dang
Université de Lorraine, LORIA, F-54506
Inria, F-54600
CNRS, LORIA, F-54506
quang-vinh.dang@inria.fr

Claudia-Lavinia Ignat
Inria, F-54600
Université de Lorraine, LORIA, F-54506
CNRS, LORIA, F-54506
claudia.ignat@inria.fr

***Abstract*—Real-time collaborative editing allows multiple users to edit a shared document at the same time. It received a lot of attention from both industry and academia and gained in popularity due to the wide availability of free services such as Google Docs. While these collaborative editing systems were initially used in scenarios involving only a small set of users such as for writing a research article, nowadays we notice a change in the scale from several users to communities of users. Group note taking during lectures or conferences is an emerging practice.**

An important measure of performance of real-time collaborative editing systems is delay. Delays exist between the execution of one user modification and the visibility of this modification to the other users. They can be caused by network physical communication media, complexity of consistency maintenance algorithms and system architecture. Some user studies have shown that delay affects group performance in collaborative editing. In this paper, we measure delays in popular real-time collaborative editing systems such as Google Docs and Etherpad and we study whether these systems could cope with large scale settings from a user perspective. Our results show that these systems are not yet ready for large-scale collaborative activities as either they reject new users connection or a high delay appears when facing an increasing number of users or their typing speeds in the same shared document.

I. Introduction

Today team working is a key role of success in companies or organizations. Very often members within an organisation or between different collaborating organisations are located at different geographical places and can work at different times. For an effective collaboration, team members usually need to use collaborative tools in order to overcome the geographical distance. Real-time collaborative editing systems are commonly used as they allow multiple users to edit a shared document at the same time.

Benefits of real time on-line collaborative editors are multiple. Firstly, they provide a ready-to-use platform for all users to view and modify documents on their web browsers, without installing heavy software bundle such as Microsoft Office or Libre Office. Secondly, they provide an environment where multiple users can contribute to shared documents in a fast and easy manner. While sharing documents by emails or physical mediums such as USB sticks would require to manually deal with multiple concurrent revisions and using version control systems such as git and svn would require trained users, in real-time collaborative editing merging is automatically performed without any user intervention.

ISBN 978-3-901882-83-8 © 2016 IFIP

Business analysis showed that new cloud collaborative editing systems such as Google Drive are taking the market share from the traditional document software provider such as Microsoft Office [1]. The number of users of Google Drive service increased from 10 millions on 2012 [2] to 240 millions on October 2014 [3]. On September 2015 Google Drive announced that there are one million paying customers using their service [4].

Initially, real-time collaborative editing systems were used in scenarios involving a small number of concurrent editing users (e.g, up to ten) such as writing a research article or a brainstorming session. However, scenarios involving a large number of concurrent editing users are emerging, such as students of a class or participants in a conference that collaboratively take notes. A recent example is a MOOC (Massive Open Online Course) where the 40,000 participants were asked to access to parts of the Google Docs documents created for the course. Due to the high number of concurrent edits to the same documents, the system crashed and finally the lecture was cancelled [5].

Various quality aspects should be taken into consideration in the design phase of large-scale collaborative editing systems. One of the important requirements of these systems is delay [6], [7]. Delays exist between the execution of one user's modification and the visibility of this modification to the other users. Delays can be caused by different reasons: network delay due to physical communication technology be it copper wire, optical fiber or radio transmission; complexity of various algorithms for ensuring consistency, where most of them depend on the number of users and number of operations that users performed; the type of architectures: For thin client architectures the computation for algorithms for maintaining consistency is done mainly on the server, which becomes a bottleneck in the case of a large number of users and operations, and therefore causes an increased delay for seeing operations of other users. For thick client architectures the computation is done mainly on the client side and delays are lower in this case.

In the context of collaborative editing, the delay is a critical

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

concern and it has a great influence on the performance and behavior of users [7]. By studying users' behavior in real-time collaborative note taking with artificial added delay, Ignat et al [6] claimed that "delay increases grammatical errors and redundancy, resulting in a decreased quality of the task content". However, delay has not been addressed at its desired level in the development of real-time collaborative editing systems.

In this paper, we aim at measuring the performance of real-time collaborative editing systems from users point of view. More precisely, we measure the perceived delay by users in online real-time collaborative editing systems in their normal working environment, i.e. using web browsers.

Setting up an experiment with numerous real users that edit concurrently a shared document would not be possible with current tools. Existing tools restrict the number of users editing a document and most of them are not open-source in order to allow code instrumentation for delay measurement. We instead simulated user behavior by means of agents that use popular web-based real-time collaborative editing services currently available in the market: Google Docs[1] and Etherpad[2].

The paper is structured as follows. We start by describing the set-up of our experiments: how we modeled and simulated user behavior and how we measured delays. We next present the performance evaluation of Google Docs and Etherpad and discuss the results obtained. We finally provide concluding remarks.

II. Evaluation settings

A. User Behavior

We define user behavior as a set of the following actions, which can be further extended:

- Start a web browser instance (Firefox, Chrome, Internet Explorer, Safari, Edge, etc.).
- Surf to the dedicated web page of a collaborative editor.
- Load a shared document.
- Perform modifications by inserting and/or deleting characters in the document.
- Interact with buttons on the web page by using mouse / pointing system.
- Close the web browser, since current collaborative editors automatically save user changes.

B. Simulation Settings

In order to simulate the real user behavior on web browsers, we selected Selenium [8], which has been widely accepted in web-based testing community [9].

The simulation is distributed on multiple computers[3]. Three types of simulated users have been defined:

1) Writer: writes a specific string to the shared document.
2) Reader: waits and reads the specific string from the writer.
3) DummyWriter: writes random strings to the shared document. Random strings are different from the specific string. DummyWriters are used to simulate concurrent users.

Each simulator (Writer, Reader, DummyWriter) performs its task on different Google Chrome browser window. The delay is measured by the time period between the moment the specific string is written by the Writer and the moment when the specific string is read by the Reader. In order to avoid clock synchronisation issues, both Reader and Writer are executed on the same computer.

C. Experiment Settings

For each real-time collaborative editing system, i.e. Google Docs and Etherpad, we measured the performance (delay) in different settings by varying the number of users who modify the document at the same time, and their typing speed, i.e. the number of characters each user types to the document in one second.

As the number of users that can concurrently modify a document in Google Docs[4] is limited to 50, we varied the number of users from one to 50. The usual range of user typing speed is 2–4 characters per second [10]. We also considered that higher speeds could be achieved by performing cut and paste operations on large blocks of text. We therefore varied the typing speed from one to 8 characters per second.

We created five shared documents and then evaluated the delays in turn on each of these documents and for each combination of settings (number of users and typing speed). In order to further eliminate random effects on the performance achieved, for each of the shared document and for each combination of settings, we repeated the experiment four times.

We used five local computers located at Inria Grand-Est, Nancy, France with the corresponding configuration features described in Table I. Clients simulating user behavior were executed on one of these computers: the Writer and Reader are executed by the first computer (with CPU Intel i7 720QM), and the DummyWriters are executed by other computers. DummyWriters are assigned to computers in a load balancing fashion: during the experiment when the number of clients, i.e. DummyWriters, is increased, each new DummyWriter is executed on the computer running the minimum number of clients with respect to its capacity, i.e. CPU and memory. We report the maximum number of DummyWriter which are executed on each computer in the third column of Table I.

III. Results

In this section, we present and discuss the performance evaluation results of two popular real-time collaborative editing systems: Google Docs and Etherpad.

[1] https://docs.google.com

[2] http://etherpad.org

[3] The implementation is available at https://github.com/vinhqdang/collaborative_editing_measurement

[4] https://support.google.com/docs/answer/2494827?hl=en as on 15-Feb-2016.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

TABLE I
THE EXPERIMENT CONFIGURATION

CPU	Memory	Number of simulated users
Intel i7 720QM	8 GB	2
Intel Xeon W3550	8 GB	15
Intel Xeon W3520	8 GB	15
Intel Core 2 Duo E6850	4 GB	10
Intel Core 2 Duo E6550	4 GB	9

A. Google Docs

Google Docs is the most popular real-time collaborative editing system today. The service was introduced in 2007, and quickly attracted over one million users [1], [2].

The results of performance evaluation of Google Docs are displayed in Figures 1–5 for different typing speeds. The delays mean value line depicted in each figure shows the increasing trend of delay with an increasing number of users that join and modify the shared documents at the same time.

The above graphs show a very interesting feature of Google Docs. When the number of users is less than ten, Google Docs provides a very good and stable performance. The delays are very small and stable meaning that the performance of the system has not been affected when the number of users increases from one to ten. However, when the number of users exceeds ten, the performance of Google Docs decreases quickly, meaning that the delay increases significantly. This might be an explanation for the limit of 50 concurrent users specified by the Google Docs documentation.

The results also show us another interesting property of Google Docs: a higher typing speed leads to a higher dependency of delay on number of users. In other words, a higher typing speed will lead to a higher delay, and the delay also increases faster with the number of users.

We notice that it is very common to observe delays over ten seconds with Google Docs. Moreover, even if Google Docs documentation claimed that up to 50 users can modify a shared document at the same time, it is not always the case. We only can simulate 50 users if the typing speed of users is one character per second. Otherwise, if we increase the typing speed, a maximum of 38 users can log in and use the service. Additional users cannot use the system as they are repeatedly displayed the following message at login "Wow this file is really popular! Some tools might be unavailable until the crowd clears".

B. Etherpad

Etherpad is a popular open-source web based collaborative platform, with the first version being released in 2008. Etherpad is currently being used by many open-source and non-profit organizations, such as Wikimedia[5].

In order to evaluate the performance of Etherpad, we installed the source code provided by Etherpad development team[6] on our own server (Intel Xeon W3550) and performed

[5]https://etherpad.wikimedia.org
[6]http://etherpad.org/\#download

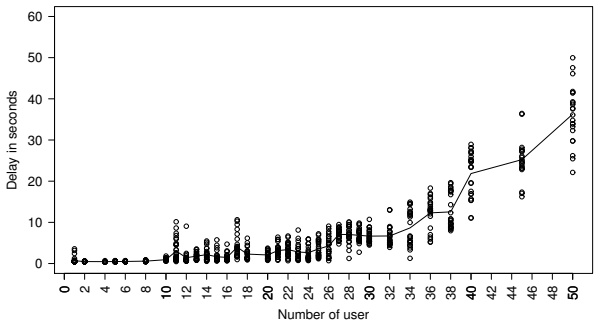


Fig. 1. Performance of Google Docs with a typing speed of one character / second

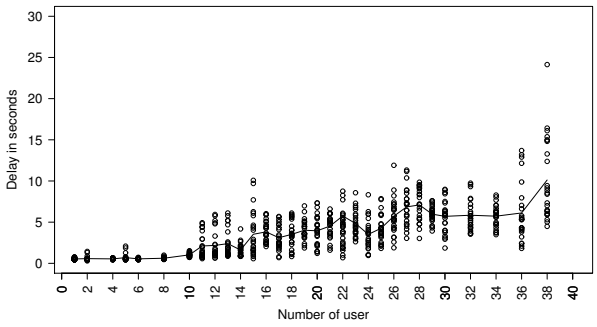


Fig. 2. Performance of Google Docs with a typing speed of two characters / second

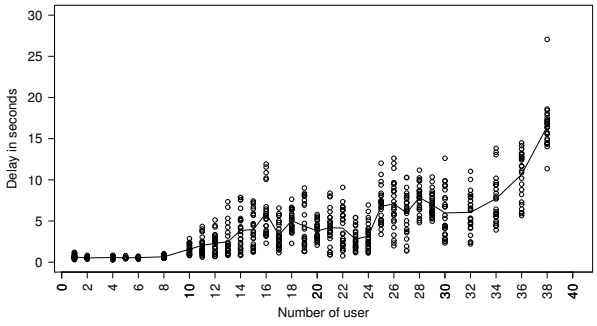


Fig. 3. Performance of Google Docs with a typing speed of four characters / second

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

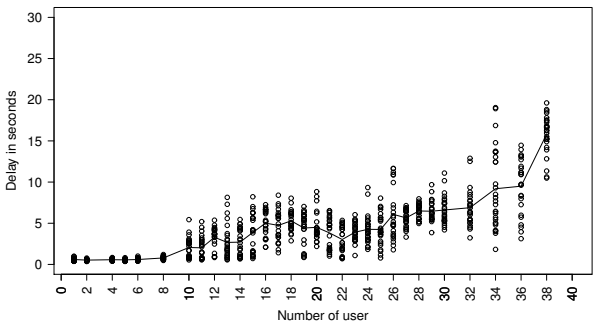


Fig. 4. Performance of Google Docs with a typing speed of six characters / second

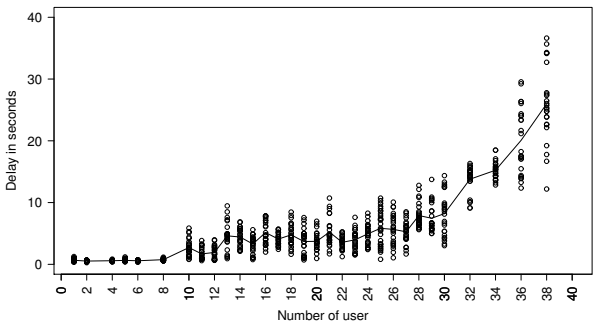


Fig. 5. Performance of Google Docs with a typing speed of eight characters / second

the same evaluation method that we previously described for Google Docs. We applied all default settings of Etherpad (i.e. we used dirtyDB as the underlying database), and maintained the same evaluation settings as in the case of Google Docs.

When the number of users is less than ten, we observe a similar phenomenon as Google Docs: Etherpad responds quickly to users' modifications, and delays are small and stable. However, when the number of users exceeds ten, Etherpad starts rejecting connections from new users, as it can be seen in the screenshot in Figure 6.

We can conclude that, Etherpad cannot be used by more than ten users that concurrently modify the shared document at the same time.

IV. Discussion

A. Source of delay

As previously mentioned, in real-time collaborative editing, delays can appear because of network due to the structure and configuration of the Internet, which basically operates on the "best effort" principle, trying to deliver data from a computer to other ones as fast as possible but without any guarantee of time bound. Delays are also due to the collaborative editing systems architecture such as client-server, peer-to-peer, thin or

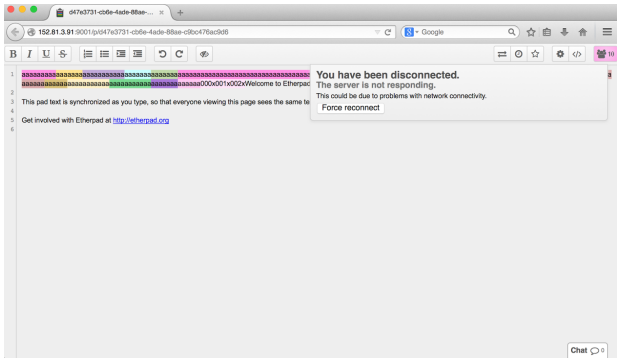


Fig. 6. Etherpad rejects users' connection when the number of users exceeds ten

thick client. Moreover, delays are due to the synchronisation algorithms implemented by each system.

In this paper, we measured the delay with different number of users and typing speeds. The delay the real users observe depends on how many users are modifying the sharing document at the same time, and how fast they are typing. However, we could not perform a more in-depth analysis on the reasons of delay as information about the inside architecture and algorithms used in Google Docs is not available and we could not take into account all the network infrastructure from our computers to Google Docs server.

During the experiments, we monitored the network bandwidth and CPU usage on our experimental computers[7]. The monitoring logs showed that our computers never consume more than 30% of CPU usage, and the incoming/outgoing network bandwidth is always less than 100 Kilobytes per second, which can be easily satisfied with the Internet connection today [11].

B. Sharing limit

Both Google Docs and Etherpad, the two popular real-time collaborative editing systems today, do not support a large number of users. Etherpad stops to accept new incoming users when the number of current users is more than ten. Google Docs claims that the service can support up to 50 users that can modify the shared document at the same time. However, this is true only if these users type at slow speed (1 character / second). Otherwise, the service cannot support more than 38 users that modify the shared document at the same time.

Therefore, Google Docs and Etherpad are not suitable for large-scale collaborative editing activities today, where hundreds of persons can share and modify a document such as a notes in events, meetings, conferences or MOOCs.

C. Effect on users

The effect of delay on users' behavior in collaborative editing has been studied in [6], [12]. Twenty groups of four students from a French university have been recruited to

[7]CPU usage and network bandwidth were monitored by using *multiload indicator*, which is available at https://launchpad.net/indicator-multiload

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

perform an experiment including several tasks in collaborative editing. Users were asked to perform each collaborative task using the Etherpad collaborative editor under instructions that demanded interleaved work. Users were allowed to coordinate themselves by using the chat available in Etherpad. Without informing participants, delays were intentionally added, i.e. the server has been programmed to wait a certain amount of time before sending data to clients. The artificial delay levels included two, four, six, eight and ten seconds, which are realistic as we discussed in Section III. Each group had to perform the required tasks under a constant but undeclared delay in the propagation of changes between group members. Software recorded each user's desktop activity, including task performance as well as chat for coordination.

The task where users were provided with a list of movies and asked to search for the release date of those movies and then sort them in an ascending order according to the release date was analysed in [12]. For the analysis of the collected data outcome metrics for measuring the quality of the realised task but also process metrics for analysing user behaviour during achievement of the tasks were introduced: a sorting accuracy based on the insertion sort algorithm, average time per entry, strategies (tightly coupled or loosely coupled task decomposition of the task), chat behavior and collisions between users. It has been found that delay slows down participants which decrements the outcome metric of sorting accuracy. Tightly coupled task decomposition enhances outcome at minimal delay, but participants slow down with higher delays. A loosely coupled task decomposition at the beginning leaves a poorly coordinated tightly coupled sorting at the end, requiring more coordination as delay increases.

The note taking task where users listened to an interview about cloud computing and took notes during this time was analysed in [6]. It has been noticed that due to delay, notes about the same topic were taken two, three and even four times. What happened is that when two users want to take notes on the same topic, in the presence of delay, changes of one user are not immediately visible to the other user, so a user thinks that the other user is not taking notes, so he/she is taking the notes. In that way, finally, the notes are in double. If more than two users try to take notes simultaneously, finally the same idea will appear three or even four times. It was found that the error rate is higher for groups that experienced a higher level of delay and redundancy is higher for groups in higher delay condition. Moreover, as delay increases the keywords depicted by users decreases. Groups were classified into two categories according to their declared experience in the domain of collaborative editing. For high experienced groups redundancy increases with the delay, but for low experienced groups the same tendency could not be observed. Chat behavior by means of number of accord words and definite determiners which together provides a common ground knowledge was considered as a measure of coordination. Low experienced groups used more coordination to manage redundancy. High experienced groups did not adjust their collaboration effort to manage redundancy.

A general hindrance of delay was observed in all analysed tasks. Delay destroys the value of collaborative editing and forces independent, redundant work.

Delays measured in GoogleDocs when the number of users exceeds ten are largely superior to the artificial delays experienced in [6], [12]. We therefore expect that a high hindrance of delay will be experienced in Google Docs in scenarios of collaborative editing that involve a large number of users that concurrently modify a shared document.

D. Implications for design

Our primary purpose is to demonstrate the delay users could observe in popular real-time collaborative editing systems available in the market, and we showed that delay is a fact in current real-time collaborative editing systems. In the scale of more than ten users to modify the sharing document at the same time, delay can come up to 50 seconds.

Usually, in distributed computing, delays could appear due to traffic congestion when the amount of data transferred between nodes overcomes the capability of the network [13]. However, as we presented in Section IV-A, we did not observe the traffic congestion during our experiments. Therefore, we could suggest that the delay mostly comes from the architecture and implementation of the services.

As claimed in the official blog of Google Drive development team [14], Google Docs is relying on the Jupiter algorithm [15] for synchronization between nodes. This might not be the best choice of algorithm to be used in collaborative editing at large scale, because the Jupiter algorithm, which belongs to Operational Transformation family of synchronisation solutions [16] requires a lot of computation on the server side, which increases the delay users observe in large-scale settings. Different algorithms such as CRDT [17], [18], standing for *Conflict-free Replicated Data Type* should achieve a better performance in large-scale collaborative editing and feature smaller delays [19]. Moreover, CRDT-based algorithms on strings should achieve better performance than character-based CRDT algorithms [20], [21].

Notifying delay to users could be implemented in real-time collaborative editing services. As suggested by several researches [6], [7], notified delay could let the users adapt their behaviors for the context. However, the delay notification has not been implemented yet in the real-time collaborative editing systems.

V. Conclusions

In this paper, we presented the performance measurement in term of delay in popular real-time editing services in the market. We demonstrated that, delay is a fact and high delay will appear and increase if there are more users joining to modify the shared document at the same time, or the users increase their typing speed. We demonstrated that the existing real-time collaborative systems are not yet ready for large-scale collaborative activities, as they reject the new users' connection if the number of users in the system increases a certain limit.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP

Delay destroys the value of collaborative editing and forces independent, redundant work. Therefore, delay should be avoided and needs more attention from development team. We discussed several strategies to tackle the problem such as using suitable consistency maintenance algorithms or notifying delay to users.

VI. Acknowledgements

This work has been carried out thanks to the support of the PSPC OpenPaaS::NG project funded by the "Investissements d'Avenir" French government program managed by the "Commissariat général à l'investissement" (CGI).

References

[1] A. Covert, "Will google docs kill off microsoft office?" 2013. [Online]. Available: http://money.cnn.com/2013/11/13/technology/enterprise/microsoft-office-google-docs/

[2] J. Crook, "Google drive now has 10 million users: Available on ios and chrome os," 2012. [Online]. Available: http://techcrunch.com/2012/06/28/google-drive-now-has-10-million-users-available-on-ios-and-chrome-os-offline-editing-in-docs/

[3] E. Protalinski, "Google announces 10% price cut for all compute engine instances, google drive has passed 240m active users," 2014. [Online]. Available: http://thenextweb.com/google/2014/10/01/google-announces-10-price-cut-compute-engine-instances-google-drive-passed-240m-active-users/

[4] B. Darrow, "Google drive claims one million paying customers, er, organizations," 2015. [Online]. Available: http://fortune.com/2015/09/21/google-drive-1m-paid-users/

[5] S. Jaschik, "Mooc mess," 2013. [Online]. Available: https://www.insidehighered.com/news/2013/02/04/coursera-forced-call-mooc-amid-complaints-about-course

[6] C.-L. Ignat, G. Oster, O. Fox, V. L. Shalin, and F. Charoy, "How do user groups cope with delay in real-time collaborative note taking," in *Proceedings of the 14th European Conference on Computer Supported Cooperative Work (ECSCW)*, 2015, pp. 223–242.

[7] I. Vaghi, C. Greenhalgh, and S. Benford, "Coping with inconsistency due to network delays in collaborative virtual environments," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, 1999, pp. 42–49.

[8] A. Holmes and M. Kellogg, "Automating functional tests using selenium," in *Proceedings of AGILE Conference (AGILE)*, 2006, pp. 270–275.

[9] R. Angmo and M. Sharma, "Performance evaluation of web based automation testing tools," in *Proceedings of the 5th International Conference Confluence, The Next Generation Information Technology Summit (Confluence)*, 2014, pp. 731–735.

[10] R. William Soukoreff and I. Scott Mackenzie, "Theoretical upper and lower bounds on typing speed using a stylus and a soft keyboard," *Behaviour & Information Technology*, vol. 14, no. 6, pp. 370–379, 1995.

[11] Akamai, "State of the internet report," 2015. [Online]. Available: https://www.akamai.com/us/en/our-thinking/state-of-the-internet-report/

[12] C.-L. Ignat, G. Oster, M. Newman, V. Shalin, and F. Charoy, "Studying the effect of delay on group performance in collaborative editing," in *Proceedings of the 6th International Conference on Cooperative Design, Visualization and Engineering (CDVE)*, 2014, pp. 191 – 198.

[13] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*. Addison-Wesley, 2007.

[14] G. D. Blog, "What's different about the new google docs: Making collaboration fast," 2010. [Online]. Available: http://googledrive.blogspot.com/2010/09/whats-different-about-new-google-docs.html

[15] D. A. Nichols, P. Curtis, M. Dixon, and J. Lamping, "High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System," in *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology (UIST)*, 1995, pp. 111–120.

[16] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 1989, pp. 399–407.

[17] S. Weiss, P. Urso, and P. Molli, "Logoot : A Scalable Optimistic Replication Algorithm for Collaborative Editing on P2P Networks," in *Proceedings of the 29th International Conference on Distributed Computing Systems (ICDCS)*, 2009, pp. 404–412.

[18] N. Preguiça, J. M. Marquès, M. Shapiro, and M. Letia, "A Commutative Replicated Data Type for Cooperative Editing," in *Proceedings of the 29th International Conference on Distributed Computing Systems (ICDCS)*, 2009, pp. 395–403.

[19] M. Ahmed-Nacer, C. Ignat, G. Oster, H. Roh, and P. Urso, "Evaluating crdts for real-time document editing," in *Proceedings of the ACM Symposium on Document Engineering (DocEng)*, 2011, pp. 103–112.

[20] L. André, S. Martin, G. Oster, and C.-L. Ignat, "Supporting Adaptable Granularity of Changes for Massive-scale Collaborative Editing," in *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2013.

[21] W. Yu, L. André, and C.-L. Ignat, "A CRDT Supporting Selective Undo for Collaborative Text Editing," in *Proceedings of the 10th International Federated Conference on Distributed Computing Techniques (DisCoTec) Distributed Applications and Interoperable Systems (DAIS)*, 2015, pp. 193–206.

ISBN 978-3-901882-84-5, IFIP Networking 2016 © 2016 IFIP