Computer Networks 70 (2014) 59–74

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Optimal and achievable cost/delay tradeoffs in delay-tolerant networks ☆

Argyrios G. Tasiopoulos [a], Christos Tsiaras [b], Stavros Toumpis [c,*]

[a] Department of Electronic and Electrical Engineering, University College London, 66-72 Gower Street, London WC1E 6EA, UK
[b] University of Zurich, Department of Informatics (IFI), CSG, Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
[c] Department of Informatics, Athens University of Economics and Business, Patision 76, 10434 Athens, Greece

A R T I C L E I N F O

Article history:
Received 31 December 2013
Received in revised form 9 May 2014
Accepted 11 May 2014
Available online 15 May 2014

Keywords:
Cost/delay tradeoff
Delay-Tolerant Networks (DTNs)
Dynamic flows
Evolving graphs
Geographic routing
Wireless networks

A B S T R A C T

Tradeoffs between the packet delivery delay and various types of packet transportation cost are a recurring theme in Delay-Tolerant Networks (DTNs). In this work we study such tradeoffs, first in a general and then in a mobile wireless setting.

In the general setting, we capture the tradeoff between the delivery delay of a packet and its transportation cost (which comprises a transmission component and a storage component) on the cost-delay plane using the Optimal Cost/Delay Curve (OC/DC), for the case when the packet follows optimal routes, and the Achievable Cost/Delay Curve (AC/DC), for the case when a specific (suboptimal) routing protocol is used.

Applying the framework of the general setting to mobile wireless DTNs, we evaluate a novel set of geographic routing protocols with delay-tolerant features against both state-of-the-art routing protocols (using their respective average AC/DCs) and also optimal routing (described in terms of the average OC/DC). Compared to the state-of-the-art protocols, our protocols are shown to achieve cost/delay tradeoffs much closer to the optimal one.

1. Introduction

In Delay-Tolerant Networks (DTNs) packet delivery delays can be large, and often comparable to the time needed for the network topology to change substantially. Although in many DTNs packet delivery delays are not tunable, quite frequently designers conscientiously decide to tolerate larger than absolutely needed delays in order to improve other performance metrics. For example, delay has been traded off with monetary costs [2], energy efficiency [3–5], and throughput [6–8].

Motivated by these specific instances of tradeoffs, in the first part of this work (Sections 2–4), we develop a *general* DTN framework for studying tradeoffs between the delay in the delivery of a packet and the associated transportation cost. The transportation cost comprises a transmission component (that accounts for the energy dissipated, the bandwidth used, the money spent, etc.) when a packet is transmitted, and a buffer storage component that represents the cost of using limited buffer spaces, security concerns, QoS and reliability constraints, etc. When a packet follows optimal routes, we characterize the tradeoff in terms of a Pareto curve on the cost-delay plane, termed the **Optimal Cost/Delay Curve (OC/DC)**. OC/DCs are defined in Section 2, and in Section 3 we show how to compute them efficiently. When realistic, suboptimal routing protocol are used instead, we characterize the tradeoff in terms of another curve on the cost-delay plane, the **Achievable Cost/Delay Curve (AC/DC)**, defined in Section 4.

☆ The work was submitted while the first author was with AUEB. A preliminary version of this work appeared in WoWMoM 2012 [1].

* Corresponding author. Tel.: +30 2190 8203551; fax: +30 210 8203551.

E-mail addresses: argyrios.tasiopoulos@ucl.ac.uk (A.G. Tasiopoulos), tsiaras@ifi.uzh.ch (C. Tsiaras), toumpis@aueb.gr (S. Toumpis).

http://dx.doi.org/10.1016/j.comnet.2014.05.006
1389-1286/© 2014 Elsevier B.V. All rights reserved.

The framework developed in the first part is very general, and can be readily adapted to many different DTN settings. As an example, in the second part of this work (Sections 5 and 6), we focus on a particularly important class of DTNs, i.e., mobile wireless DTNs where the changes in the topology are due to node mobility. In this setting, we design novel routing protocols that rely on geographic forwarding as well as extended sojourns of packets on node buffers. These, along with a model about the network, are presented in Section 5 and evaluated against the state of the art in Section 6, using the tools developed in the first part.

We briefly discuss related work in Section 7 and conclude in Section 8.

2. Optimal cost/delay curves

For simplicity, we assume that all packet transmissions are instantaneous and propagation delays are zero. This simplifies the analysis and is quite reasonable in our setting. Indeed, in DTNs packet delays are comparable to the time needed for the network topology to change, with respect to which transmission and propagation delays are negligible.

The network comprises a set $\mathcal{N}$ of N nodes, indexed by $1, 2, \ldots, N$. We model the fluctuating connectivity of the nodes in the continuous time interval $[0, \infty)$ as follows: node i can send a data packet to node j with a **transmission cost** $c_{ij}(t) \in [0, \infty]$. The cost $c_{ij}(t)$ is defined for all pairs (i, j), with $c_{ij}(t) = \infty$ signifying that communication from i to j is impossible at time t. Also, a packet stored at node i from time a to time b incurs a cost $\int_a^b s_i(t)dt$ where $s_i(t)$ is the **storage cost rate**, measured in units of cost over time.

The type of the communication medium as well as the nature of the transmission cost and the storage cost rate are not defined in this part of the work, in order to keep the formulation as general as possible. We briefly note that the details of the two types of cost (i.e., transmission and storage) depend on the particular network at hand. For example, in networks where energy is limited, the transmission cost could be the energy dissipated for the transmission; in networks where bandwidth is limited, the transmission cost could model the amount of interference the transmitter is causing. Regarding the storage cost, it naturally captures the cost of occupying buffer spaces, in particular when buffer capacity is limited. In cases where some nodes are more suitable than others as relays (due to security concerns, existence of altruism/selfishness, QoS requirements, a non-uniform traffic load in the network, etc.) the storage cost rate can be differentiated among nodes to steer packets as desired. The formulation is general enough to allow both cost components to be further divided into subcomponents, each associated with a different design concern (energy, bandwidth, buffer space, etc.).

Moving on to routing, let a **journey** be a *finite* sequence of alternating (i) sojourns of *finite* duration of a packet at the buffers of nodes and (ii) hops of that packet between pairs of nodes. The **starting node** of a journey $\mathbf{J}$ is denoted by $s(\mathbf{J})$, its **starting time** by $t_s(\mathbf{J})$, its **ending node** by $e(\mathbf{J})$, and its **ending time** by $t_e(\mathbf{J})$.

Consider the journey $\mathbf{J}$ of H hops of a packet that starts at node i_0 and at time t_0, the packet is transmitted by node i_{h-1} and received by node $i_h \neq i_{h-1},\ h = 1, \ldots, H$, at time t_h, and finally the packet stays at i_H from the time instant t_H of its last transmission until the time instant t_{H+1}, when the journey completes. Clearly, $t_{h-1} \leqslant t_h$, where $h = 1, \ldots, H+1$. If $t_{h-1} < t_h$, then the packet waits at node i_{h-1} for time $t_h - t_{h-1}$, otherwise node i_{h-1} relays the packet the moment it receives it. Let

$$c(\mathbf{J}) = \sum_{h=1}^{H+1} \int_{t_{h-1}}^{t_h} s_{i_{h-1}}(t)dt + \sum_{h=1}^{H} c_{i_{h-1}i_h}(t_h) \tag{1}$$

be the **(total) cost** of $\mathbf{J}$. The first term is the **storage cost** of $\mathbf{J}$, and the second term is the **transmission cost** of $\mathbf{J}$.

The **Optimal Cost/Delay Curve (OC/DC)** $C_{ij}(t)$ between nodes i and j is defined as the minimum total cost over all journeys of a packet from i to j that begin at time 0 and end by time t *the latest*:

$$C_{ij}(t) = \min_{\{\mathbf{J} : s(\mathbf{J}) = i, e(\mathbf{J}) = j, t_s(\mathbf{J}) = 0, t_e(\mathbf{J}) \leqslant t\}} c(\mathbf{J}). \tag{2}$$

Observe that there exist always journeys in (2) to minimize over—in the worst case, the minimum will be infinite.

Note that the OC/DC $C_{ij}(t)$ is a non-increasing (but not necessarily strictly decreasing) function of t. Indeed, increasing time t means that the minimum of (2) is taken over a larger set of journeys. Therefore, the OC/DC $C_{ij}(t)$ captures the fundamental tradeoff between delay and cost that exists in *all* DTNs, i.e., increasing the time delay tolerance can only reduce the incurred cost.

The tradeoff captured by the OC/DC can only be achieved by the nodes if they are aware of the complete topology evolution of the network. Indeed, let $\mathbf{J}_0$ be an optimal journey that achieves the minimum $C_{ij}(t)$, for some time $t > 0$. In order to compute the part of $\mathbf{J}_0$ until time $t_j < t$, the evolution of the network in the *future* interval $[t_j, t]$ is also needed. For example, in a mobile wireless setting, if, at a given point in time, a packet is faced with a choice about which of two nodes to sojourn on, the optimal decision depends on the future trajectories of both these nodes. The evolution of the network is indeed known in advance in some notable cases, for example Space DTNs [9] and Vehicular DTNs comprised of buses moving along fixed schedules [10].

Nevertheless, it is also very useful to calculate the OC/DC even in settings where the future (or even the past) evolution of the network is not completely available to the nodes at any time instant. Indeed, knowing the OC/DC provides an absolute yardstick with which the performance of practical protocols can be compared. What is more, this yardstick is highly informative, as it is not a single number but rather a Pareto curve. In Section 6 we employ OC/DCs in this manner.

Finally, we define the **Punctual Cost/Delay Curve (PC/DC)** $P_{ij}(t)$ between nodes i and j as the minimum cost with which a packet can leave node i at time 0 and exist at node j at *exactly* time t:

$$P_{ij}(t) = \min_{\{\mathbf{J} : s(\mathbf{J}) = i, e(\mathbf{J}) = j, t_s(\mathbf{J}) = 0, t_e(\mathbf{J}) = t\}} c(\mathbf{J}). \tag{3}$$

The optimization is over all journeys from node i to node j that begin at time 0 and end at exactly time t. For some of these journeys, the last hop (to node j) took place at some time $t' < t$, and so the cost of the sojourn during the time interval $[t', t]$ at node j is included in $c(\mathbf{J})$.

PC/DCs come handy as the minimum cost with which a packet can be transported to a node j that is not its final destination, by time t, in anticipation of another journey that starts from j at time t (see Section 5.2). PC/DCs are also used in the efficient calculation of OC/DCs in Section 3.

3. Efficient computation of OC/DCs

3.1. Cost/delay evolving graph

Let $[0, T]$, with T an integer, be a time interval over which we would like to compute the OC/DCs $C_{ij}(t)$ for all pairs of $i,\ j$. We divide this interval into T **epochs** $[k-1, k]$ of unit duration, where $k = 1, 2, \ldots, T$. We refer to the instances $0, 1, 2, \ldots, T$ as the **epoch transition times**.

To facilitate computations, we assume that packet transmissions take place exclusively during epoch transition times, and, as transmissions are instantaneous, a packet can be transmitted during an epoch transition an arbitrary number of times.[1] Secondly, during each epoch, the values of the storage cost rates remain fixed; changes in these values are allowed only during epoch transitions. Finally, we only evaluate the OC/DCs during epoch transition times.

Intuitively, epochs are periods of time during which we expect the DTN to not change significantly. Increasing the number of epochs that span a given time period improves the accuracy of our discrete time model. (Note that, by a rescaling of the time axis, it is always possible to assume that the duration of all epochs is a unit of time.) However, note that for the above assumptions to hold, the epoch duration should always be much larger than the time it takes a packet to be transmitted.

Slightly abusing notation, in this section we let the time index t, so far continuous, take the discrete values $0, 1, 2, \ldots, T$. We keep the same notation for the discretized versions of $C_{ij}(t),\ P_{ij}(t),\ c_{ij}(t)$, and $s_i(t)$.

The time discretization allows the depiction of the evolution of the network using a static graph $\mathcal{G}$ termed the **Cost/Delay Evolving Graph (C/DEG)**.[2] $\mathcal{G}$ comprises a cascade of $T+1$ **replicas** $\mathcal{G}^t = (\mathcal{N}^t, \mathcal{A}^t),\ t = 0, 1, \ldots, T$, each replica corresponding to a single epoch transition. We set $\mathcal{N}^t = \{1^t, 2^t, \ldots, N^t\}$, i.e., there is a **node replica** for each of the nodes in the network. The arc set $\mathcal{A}^t$ of replica t contains a **link arc** (i^t, j^t) for each set of distinct nodes $i,\ j$ for which $c_{ij}(t) < \infty$, with cost $c_{ij}(t)$. Let A^t be the number of link arcs in $\mathcal{A}^t$. Finally, we connect consecutive replicas t and $t+1$, where $t = 0, 1, \ldots, T-1$, with **storage arcs** (i^t, i^{t+1}), for all $i = 1, \ldots, N$, with cost $s_i(t)$. As an example, in Fig. 1 we plot a simple C/DEG of a network of $N = 4$ nodes and $T+1 = 3$ replicas.

Observe then that each journey is uniquely represented by a single path in the C/DEG, and vice versa. For example, in the context of Fig. 1, consider the journey of a packet that goes from node 1 to node 2 and then to node 3 at time 0, sojourns at node 3 for epoch $[0, 1]$, is then transmitted to node 2 and then to node 4 at time 1, sojourns at node 4 during epoch $[1, 2]$, and is finally transmitted to node 3 at time 2. The corresponding C/DEG path is

$(1^0, 2^0, 3^0, 3^1, 2^1, 4^1, 4^2, 3^2)$.

Furthermore, the journey cost (1) is equal to the cost of the related C/DEG path.

Hence, it follows that the PC/DC value $P_{ij}(t)$ equals the minimum cost among the costs of all paths from i^0 to j^t. Also, the OC/DC value $C_{ij}(t)$ equals the minimum cost among the costs of all paths starting at i^0 and ending at any of the nodes $j^0, j^1, \ldots, j^t$. Therefore, for any $i,\ j,\ t$,

$$C_{ij}(t) = \min_{k=0,\ldots,t} P_{ij}(k). \tag{4}$$

3.2. Efficient OC/DC calculation

We now introduce algorithms for computing efficiently the values $C_{ij}(t)$ and their associated optimal paths $c^{i \to j}(t)$, as well as the values $P_{ij}(t)$ and their associated optimal paths $p^{i \to j}(t)$.

Our algorithms require solving single-source shortest path problems for finding the shortest paths from one source node to all other nodes in a graph. We use Dijkstra's algorithm with a Fibonacci heap implementation, whose complexity is $O(A + V \log V)$ where A and V are the numbers of arcs and vertices [12]. Henceforth, we refer to this algorithm as Dijkstra.

We assume that the number of arcs in each replica A^t is upper bounded by $k_1 N \log N$ where k_1 is a constant that can be arbitrarily large, but is not a function of T or N. The latter bound is guaranteed if the degrees of all replica nodes are bounded by $k_1 \log N$. (Note that two nodes i^t and j^t are connected with an arc only if $c_{ij}(t) < \infty$.)

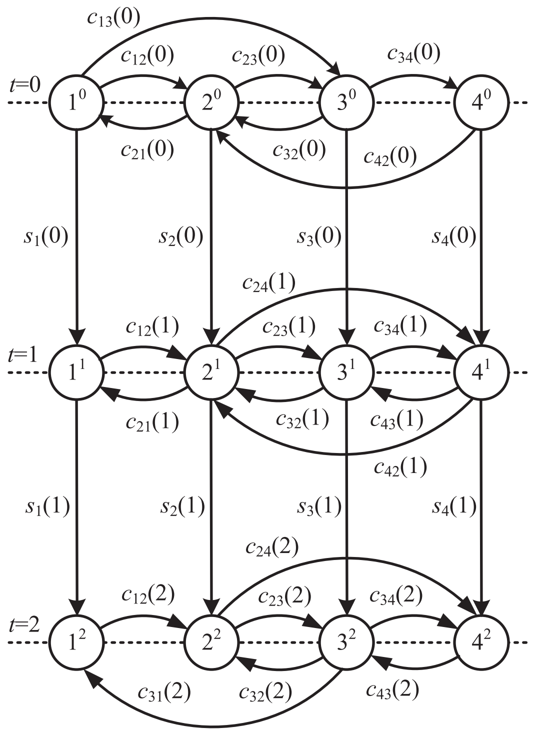

Fig. 1. Example C/DEG with 4 nodes and 3 replicas.

[1] Limiting the number of transmissions during an epoch transition can be integrated in our formulation in a straightforward manner, but we omit this for simplicity.

[2] Our term follows the term *Evolving Graph* for a related graph in [11].

Algorithm 1. Baseline Calculation of OC/DCs.

```
1  for $i = 1$ to $N$ do
2      Find $P_{ij}(t)$ and $p^{i \to j}(t)$ for all $j, t$, by executing Dijkstra on the C/DEG with source $i^0$;
3  end
4  for $i = 1$ to $N$ do
5      for $j = 1$ to $N$ do
6          $C_{ij}(0) := P_{ij}(0)$;    $c^{i \to j}(0) := p^{i \to j}(0)$;
7          for $t = 1$ to $T$ do
8              $C_{ij}(t) := \min\{C_{ij}(t-1), P_{ij}(t)\}$;
9              if $C_{ij}(t) = C_{ij}(t-1)$ then
10                 $c^{i \to j}(t) := c^{i \to j}(t-1)$;
11             else
12                 $c^{i \to j}(t) := p^{i \to j}(t)$;
13             end
14         end
15     end
16 end
```

Algorithm 1 does not make use of the special structure of the C/DEG, and so represents a baseline algorithm. In the first **for** loop (lines 1–3), all PC/DCs, and their respective optimal paths, are calculated by running Dijkstra N times. In the second (outer) **for** loop (lines 4–16), the OC/DCs, and their respective optimal paths, are calculated using an iterative form of (4).

The computation cost of line 2 is $O(A + V \log V)$ where $A = (T+1)O(N \log N)$ and $V = (T+1)N$. The computation cost of lines 8–13 is 1, and therefore the computational cost of Algorithm 1 is $O(TN^2 \log(TN))$.

Algorithm 2 makes use of the special structure of the C/DEG to expedite calculations. Observe that the only arcs coming out of replica t, where $t = 0, \ldots, T-1$, are all those of the form (i^t, i^{t+1}), where $i = 1, \ldots, N$. Therefore, to calculate the PC/DCs, instead of executing Dijkstra on the complete C/DEG once for each source node i, as we did in Algorithm 1, we can execute Dijkstra T times for that node, once for each replica, and each time on essentially that single replica.In more detail, in each execution of the first outer **for** loop (lines 1–17) we calculate $P_{ij}(t)$ and $p^{i \to j}(t)$ for a specific source node i, and for all j, t. Line 2 is an initialization, wherein $P_{ij}(0)$ and $p^{i \to j}(0)$ are found, using only replica.

In each of the T executions of the middle **for** loop (lines 3–16) we calculate $P_{ij}(t)$ and $p^{i \to j}(t)$ for all j and for one value of t. In lines 4–6 we create a **virtual graph** $\mathcal{G}^v$ that comprises

Algorithm 2. Fast calculation of OC/DCs.

```
1  for $i = 1$ to $N$ do
2      Find $P_{ij}(0)$, $p^{i \to j}(0)$ for all $j$, by executing Dijkstra on $(\mathcal{N}^0, \mathcal{A}^0)$, with source $i^0$;
3      for $t = 1$ to $T$ do
4          $\mathcal{G}^v := (\mathcal{N}^t, \mathcal{A}^t)$;
5          Augment $\mathcal{G}^v$ by adding $I$ and $N$ arcs $(I, j^t)$;
6          Set the cost of $(I, j^t)$ to $P_{ij}(t-1) + s_j(t-1)$ for all $j$;
7          Find $P_{ij}(t)$ and associated optimal paths $v^{i \to j}(t)$ for all $j$ by executing Dijkstra on
           $\mathcal{G}^v$ with source $I$;
8          for $j = 1$ to $N$ do
9              if $v^{i \to j}(t) = (I, j^t)$ then
10                 $p^{i \to j}(t) := (p^{i \to j}(t-1),\ j^t)$;
11             else
12                 Let $v^{i \to j}(t)$ be $v^{i \to j}(t) = (I,\ k^t,\ q,\ j^t)$ /* $q$ is some path of replica nodes */ ;
13                 $p^{i \to j}(t) := (p^{i \to k}(t-1),\ k^t,\ q,\ j^t)$;
14             end
15         end
16     end
17 end
18 Execute lines 4-16 of Algorithm 1 to calculate $C_{ij}(t)$, $c^{i \to j}(t)$ for all $i, j, t$;
```

Algorithm 3. Parallel calculation of OC/DCs.

```
for $t = 0$ to $T$ do
    for $i = 1$ to $N$ do
        Calculate costs $R_{ij}(t)$ and paths $r^{i\to j}(t)$ for all $j$ by executing DIJKSTRA in $(\mathcal{N}^t, \mathcal{A}^t)$;
    end
end
for $i = 1$ to $N$ do
    for $j = 1$ to $N$ do
        $P_{ij}(0) := R_{ij}(0)$;    $p^{i\to j}(0) := r^{i\to j}(0)$;
        $C_{ij}(0) := R_{ij}(0)$;    $c^{i\to j}(0) := r^{i\to j}(0)$;
    end
end
for $t = 1$ to $T$ do
    for $i = 1$ to $N$ do
        for $j = 1$ to $N$ do
            $w := \arg\min_{k=1,\dots,N} [P_{ik}(t-1) + s_k(t-1) + R_{kj}(t)]$;
            $P_{ij}(t) := P_{iw}(t-1) + s_w(t-1) + R_{wj}(t)$;
            $p^{i\to j}(t) := (p^{i\to w}(t-1),\ w^t,\ r^{w\to j}(t))$;
            Execute lines 8-13 of Algorithm 1 to find $C_{ij}(t),\ c^{i\to j}(t)$;
        end
    end
end
```

the replica t along with a single **virtual node** I with arcs to each node j^t. In line 6 we set the cost of each of these arcs equal to the cost incurred to a packet traveling optimally to j^{t-1} and spending epoch $[t-1, t]$ at node j. In line 7 we execute DIJKSTRA on $\mathcal{G}^v$ with source I, thus finding the optimal paths $v^{i\to j}(t)$ and costs from I to each j^t, *on the virtual graph*. The optimal costs are clearly also the values of $P_{ij}(t)$. In lines 8–15 we combine the optimal paths $v^{i\to j}(t)$ and the previously found optimal paths $p^{i\to j}(t-1)$ to calculate the optimal paths $p^{i\to j}(t)$ *on the initial C/DEG*. Finally, in line 18, we execute the second outer **for** loop of Algorithm 1 to find the OC/DCs $C_{ij}(t)$ and the associated $c^{i\to j}(t)$.

The computation cost of the first outer **for** loop (lines 1–17) is on the order of $N(T+1)$ times the cost of running DIJKSTRA which, according to our assumptions, is $O(A + V\log V)$ where $A = O(N\log N)$ and $V = N+1$. Therefore, this is $O(TN^2\log N)$. As the cost of the second outer **for** loop (line 18) is $O(TN^2)$, it follows that the total cost is $O(TN^2\log N)$.

With respect to Algorithm 1, the computation cost is reduced by a factor of $\log T$, which is modest. The gain in memory is by a factor of T, which is more substantial: using Algorithm 1 we need to store simultaneously $T+1$ replicas. With Algorithm 2, we only use one replica at the time.

Both Algorithms 1 and 2 can be readily parallelized, using up to N processors, as the values $C_{ij}(t)$ can be calculated independently for each node i. The next algorithm, Algorithm 3, can employ more than N processors, and so, provided we have enough of them, is faster.

Algorithm 3 operates as follows. Let $R_{ij}(t)$ be the minimum cost with which node i can send a packet to node j using a zero-duration journey during epoch transition time t. Let $r^{i\to j}(t)$ be the associated path. The first outer **for** loop (lines 1–5) calculates $R_{ij}(t)$ and $r^{i\to j}(t)$ for all $i,\ j,\ t$. Lines 6–11 are used to initialize the values of $P_{ij}(t)$ and $C_{ij}(t)$ and their associated paths for $t = 0$. Finally, in lines 12–21 we calculate $P_{ij}(t),\ C_{ij}(t)$ and their associated paths for $t = 1, \dots, T$.

Regarding the execution time, in the first outer **for** loop (lines 1–5) we execute DIJKSTRA TN times on graphs with N vertices and $O(N\log N)$ arcs, therefore the number of calculations needed is $TNO(O(N\log N) + N\log N)$. Since these TN operations can be executed independently, the execution time, assuming we have p processors, is

$$O((TN^2\log N)/p_1), \quad p_1 = \begin{cases} TN, & p \geqslant TN, \\ p, & p < TN. \end{cases} \tag{5}$$

The execution time of the second outer **for** loop (lines 6–11) is smaller than the time needed for the first outer **for** loop, and so can be ignored. Regarding the third outer **for** loop (lines 12–21), lines 15–18 are executed TN^2 times. The costliest operation is the minimization over N numbers of lines 15–16. However, assuming we have at least N processors, the execution time for this minimization is $O(\log N)$. Indeed, we can place these numbers at the leaves of a full binary tree of depth $O(\log N)$ and perform $O(\log N)$ batches of simultaneous pairwise comparisons, at the end of each batch placing the resulting minima one level up the tree. Noting that the two inner **for** loops are fully parallelizable, it follows that the total execution time for lines 12–21 is

$$O((TN^2\log N)/p_2),\ p_2 = \begin{cases} N^2, & p \geqslant N^3, \\ p/N, & N \leqslant p < N^3. \end{cases} \tag{6}$$

Summing (5) and (6), we find the total execution time. As a limiting case, if we have $\max\{TN, N^3\}$ processors, then the total execution time is $O((T+N)\log N)$, which is notably better than the time $O(TN\log N)$ needed by Algorithm 2 when it employs its maximum of N processors.

4. Achievable cost/delay curves

We define a **routing protocol** P to be a rule that, given a packet located at node i at time 0, a destination j for that packet, and a **parameter vector** $\mathbf{p}$, creates a journey $\mathbf{J}(i \to j, \mathbf{p})$. The parameter vector $\mathbf{p}$ comprises the values of possibly many tunable parameters that affect the operation of the rule. For example, in the case of wireless DTNs, two tunable parameters could be the transmitter power and the angular width of the radiation pattern of the directional antennas used, therefore $\mathbf{p} \in [0, P_{\max}] \times [0, 2\pi]$. Tunable parameters provide the rule with degrees of freedom that can be used in trading off delay with cost. The parameter vector takes its values from a **parameter space** $\mathcal{P}$. If no tunable parameters exist, then the parameter space reduces to a unit set $\mathcal{P} = \{\mathbf{p}_0\}$.

A few comments are in order: firstly, the definition includes protocols that come up with journeys that fail to send the packet to j, or have infinite cost. Secondly, as by definition all journeys have a finite duration, the routing protocols is expected to include mechanisms that guarantee the eventual termination of the packet forwarding, using e.g. Time-To-Live (TTL) counters. Thirdly, we leave no room for randomness in our formulation, and the journey is always deterministically specified by the protocol P given the parameter vector $\mathbf{p}$. Finally, the protocol may make full use of the information in the network model $\mathcal{N}$ or it may only use part of the information, such as locally available information about the present and past topology.

Let the **Achievable Cost/Delay Curve (AC/DC)** $A_{ij}(t)$ between nodes i, j under protocol P be defined as the minimum cost, over all possible parameter settings, with which a packet can travel from node i at time 0 to node j by time t the latest:

$$A_{ij}(t) = \min_{\{\mathbf{p}\,:\,t_e(\mathbf{J}(i\to j,\mathbf{p})) \leqslant t,\, e(\mathbf{J}) = j\}} c(\mathbf{J}(i \to j, \mathbf{p})). \tag{7}$$

If there is no parameter setting for which the resulting journey ends at j by time t the latest, then the minimum is over an empty set and is taken to be ∞.

When the protocol has no tunable parameters, i.e., $\mathcal{P} = \{\mathbf{p}_0\}$, the definition simplifies to

$$A_{ij}(t) = \begin{cases} c(\mathbf{J}_0), & t \geqslant t_e(\mathbf{J}_0), e(\mathbf{J}_0) = j, \\ \infty, & \text{otherwise.} \end{cases}$$

where $\mathbf{J}_0 = \mathbf{J}(i \to j, \mathbf{p}_0)$.

As with the OC/DC $C_{ij}(t)$, the AC/DC $A_{ij}(t)$ captures the cost/delay tradeoff of DTNs, but for a specific routing protocol P: if we can tolerate a delay of at most t, then there is a combination of parameters $\mathbf{p}$ such that the routing protocol can route the packet to j with cost equal to $A_{ij}(t)$; if we increase our delay tolerance t, then the protocol can deliver the packet with a cost no greater than in the previous case, and possibly smaller, by making use of the extra time afforded to it and the associated opportunities arising from the evolution of the topology, and by choosing its parameters accordingly.

The usefulness of the AC/DC lies in the detailed picture of the cost/delay performance of protocol P it provides. Drawing the AC/DCs of different routing protocols allows a more in-depth comparison than the comparison using scalar figures of merit; drawing an AC/DC in the same plot with the corresponding OC/DC allows the evaluation of P in absolute terms.

5. Delay-tolerant geographic routing

We now apply the general framework we developed in Sections 2–4 to the specific case of mobile wireless DTNs that use a family of hybrid delay-tolerant/geographic routing protocols.

We augment the network model of Section 2 so that it applies to mobile wireless environments. In particular, when node i transmits a packet across a distance d, the cost incurred is $c(d)$, where $c(\cdot)$ is the **distance cost function**. Also, $c(d) = \infty$ iff $d > R_{\max}$, where $R_{\max}$ is the **maximum communication range**. Let the **neighborhood** $O_A(t)$ of a node A be a circle centered at all times at node A, with radius $R_{\max}$. The nodes in $O_A(t)$, called the **neighbors** of A, can directly communicate with A with finite cost.

5.1. Location information

To make a routing decision at time t_0, each node A requires information on the future movement of all nodes in its neighborhood $O_A(t_0)$ as well as all destinations of packets in its queue. In more detail, for each node B that falls into any of these two node groups, node A requires an estimation of its **trajectory** $\{\mathbf{r}_B(t) : t_0 \leqslant t \leqslant t_0 + T_e\}$, where $\mathbf{r}_B(t)$ is the location of B at time t, and T_e is a global parameter that specifies how much in the future extend the trajectory estimations.

In practice, T_e should be comparable to the expected delivery delays of the packets. The estimations do not have to be perfect: in Section 6 we examine the effects of knowing future trajectories for only a limited amount of time in the future and of estimating them wrongly. If no complete trajectory estimation can be made, we also present, in Section 5.4, two protocols with more modest requirements.

As a first step towards fulfilling this requirement, it is necessary for each node to be able to estimate its own future trajectory. In many cases, as in vehicles equipped with automotive navigation systems that plan the routes taken, nodes have ready access to trajectory estimations. Otherwise, provided a node knows its location at all times (using some positioning technique) and records it, it can estimate its future trajectory based on its past trajectory and/or history of trajectories; if the mobility is periodic or otherwise structured, the estimation can be very accurate.

The trajectory can be represented in terms of a list of waypoints along with the times the node estimates it will reach them. Hence, the location of the node at any time can be approximated by interpolation. This list is exchanged between neighbors, therefore, one would like to limit its size (trading perhaps its accuracy) and the associated communication overhead (privacy and selfishness concerns are also a reason to limit its accuracy).

While two nodes are neighbors, they will keep exchanging their estimations of their own trajectories. As this exchange will happen only when a trajectory estimation is updated, and the size of the trajectory information is expected to be modest, the control overhead will be manageable, unless the node density is too large. If the control overhead due to this exchange is excessive, appropriate measures, which fall beyond the scope of this work, should be taken, such as a coarser exchange or exchange with selected neighbors.

As discussed, the trajectory of a node must also be available to (possibly distant) nodes forwarding packets for that node. This can be achieved in practice by the sources of packets appending to them trajectory estimations of the destinations obtained from a location service [13]. Nodes further down the forwarding path may use these estimations or choose to update them.

5.2. Forwarding potential

Given a packet with destination D buffered in A at time t_0, the **Forwarding Potential (FP)** $\phi(B)$ of a node B in the neighborhood $O_A(t_0)$ of A quantifies the incentive of forwarding that packet to node B:

$$\phi(B) = \max_{\{t_1,t_2 : t_0 \leqslant t_1 \leqslant t_2 \leqslant t_0 + T_e\}} \frac{|AD(t_0)| - |BD(t_2)|}{P_{AB}(t_1) + \int_{t_1}^{t_2} s_B(t)dt + a(t_2 - t_0)}. \tag{8}$$

We take it that node A belongs to $O_A(t_0)$, therefore $\phi(A)$ is also defined, and describes how advantageous it is for the packet to remain at A.

We now clarify the role of all quantities in (8). Time $t_1 \geqslant t_0$ is the time at which the packet will arrive at node B, assuming B is chosen as a relay. Time t_2 is when node B is evaluated as a potential relay, in terms of the fraction appearing in (8) it achieves. We always have $t_2 \geqslant t_1$; if $t_2 > t_1$, the evaluation of node B is based on a time after the arrival of the packet at B.

Regarding the numerator, $|AD(t_0)|$ is the distance between nodes A and D at time t_0 and $|BD(t_2)|$ is the distance between nodes B and D at time t_2. The difference $|AD(t_0)| - |BD(t_2)|$ is the **progress** made in covering the distance from A to D in the interval $[t_0, t_2]$.

Regarding the denominator, $P_{AB}(t_1)$ is the PC/DC value corresponding to the node pair A, B, at time t_1, taking t_0 as the start of time, using nodes in $O_A(t_0)$ and their available trajectory estimations. Therefore, $P_{AB}(t_1)$ expresses the cost incurred starting on an optimal journey at time t_0 and node A and ending at time t_1 and node B, using nodes in $O_A(t_0)$. Note that if the packet arrives at node B earlier than time t_1, then there will be a storage cost incurred at node B, as B is not the final destination of the packet. It is also possible that the packet will leave node A at some time $t > t_0$, but obviously $t \leqslant t_1$. $P_{AB}(t_1)$ is calculated by node A using the algorithms of Section 3. Moving on to the next term, the integral $\int_{t_1}^{t_2} s_B(t)dt$ is the storage cost incurred at node B due to the sojourn of the packet there from time t_1 to time t_2. Finally, the time difference $t_2 - t_0$ is equal to the delay incurred until the progress of the numerator is realized. Parameter a is a positive real number used for tuning the relative weight of the delay versus the transmission and storage costs. When large, the priority is in keeping the delay $t_2 - t_0$ small, which expedites the forwarding of the packet, at the expense of increased transmission cost. If a is set to a small value, cost minimization is preferable, possibly at the cost of large delays. As a is measured in units of cost over delay, we call it the **Cost/Delay (C/D) coefficient**.

Putting everything together, the fraction in (8) represents the reduction of the distance remaining to the destination, if B is selected as the next hop, over a combination of the cost and the delay incurred at the time interval $[t_0, t_2]$ over which the progress is achieved. Maximizing over both t_1 and t_2 means selecting (i) the most favorable time t_1 for the packet to arrive at B and (ii) the most favorable time t_2 for calculating the ratio of the progress achieved over the combination of the cost and the delay incurred.

In the case of $\phi(A)$, the numerator and the denominator of (8) become 0 for $t_1 = t_2 = t_0$. To avoid this, when performing the optimization we require $t_0 < t_1 \leqslant t_2$.

We note that the distances appearing in the numerator of (8) can be Euclidean; this is appropriate in homogeneous wireless networks, as its reduction represents real progress towards the packet delivery. However, when the network is not homogeneous, for example when there are large physical areas always devoid of nodes, routing along straight lines might not be optimal or might fail altogether; hence the distance to the destination must be measured differently. For example, using the massive network framework in [14], distances must be measured along a continuous geodesic line to the destination. As another example, in a VANET setting, the distance used should be a metric taking into account that the routes taken by the packets and the vehicles must be along existing roads.

We assume that node A is aware of the storage cost rate $s_B(t)$, for $t \in [t_0, t_0 + T_e]$, of any node B in its neighborhood $O_A(t_0)$. Together with the assumptions of Section 5.1, this means that A can calculate the forwarding potentials $\phi(B)$ of any neighbor B.

5.3. General Rule

We now specify our first rule for performing packet forwarding based on FP. As later we study two simplifications of the rule, we call it the **General Rule (GR)**.

GR works as follows: let node A have a packet destined for node D. Node A continuously calculates the forwarding potential of all its neighbors, including itself, and keeps track of the maximum value found. At any time t_0 that this value corresponds to a node B other than A, *and* the optimal journey **J** corresponding to the PC/DC value $P_{AB}(t_1)$ of the denominator of (8) involves an *immediate* hop to a node C at time t_0, then node A transmits the packet to node C, and the process starts over. Due to the potential for information mismatch between neighbors, infinite routing loops are a possibility which we eliminate by not allowing nodes to transmit a packet to a node C if that packet has been at node C within the time period $[t_0 - T_1, t_0]$, where t_0 is the current time and T_1 is a global parameter. As the rule is applied afresh at each node receiving the packet, it is possible that the next hop will not be the one specified in **J** found by A.

5.4. Simplifications

To find the forwarding potential of a node B, node A must perform a double maximization over the two time instants t_1 and t_2, using extensive information about the future topology of the network. Therefore, GR takes a very informed decision, at the expense of both a lot of control traffic (needed for node A to collect the information), as well as computing resources. With this as a motivation, we now proceed to define two simplifications of GR, that have fewer requirements.

Substituting, in (8), $P_{AB}(t_1)$ with the cost $C_{A\to B}(t_0)$ of the smallest-cost *zero-duration* journey that starts and ends at time $t_1 = t_0$ results in an FP that is much simpler to calculate, because $C_{A\to B}(t_0)$ is easier to compute than $P_{AB}(t_1)$, and because the maximization is now over a single parameter. The new FP becomes

$$\phi(B) = \max_{\{t_2 : t_0 \leqslant t_2 \leqslant t_0 + T_e\}} \frac{|AD(t_0)| - |BD(t_2)|}{C_{A\to B}(t_0) + \int_{t_0}^{t_2} s_B(t)dt + a(t_2 - t_0)}.$$

Calculating the maximum over all $t_2 \in [t_0, T_e]$ might still be computationally intensive in certain applications. One solution is to set the time t_2 equal to that time within the interval $[t_0, t_0 + T_e]$ at which the numerator is maximized, i.e., the time when B is closest to D. Then,

$$\phi(B) = \frac{|AD(t_0)| - |BD(t_2)|}{C_{A\to B}(t_0) + \int_{t_0}^{t_2} s_B(t)\,dt + a(t_2 - t_0)}$$

where

$$t_2 = \arg\min_{[t_0, t_0+T_e]} |BD(t)|. \tag{9}$$

Observe that in this case nodes do not need to share (or be able to estimate) their complete future trajectory information within the time interval $[t_0, t_0 + T_e]$, but only the point in time and space where they lie closest to the destination D, reducing thus the message exchange overhead and the computations required. As this rule strives to achieve a balance between cost and delay, we call it the **Balanced Ratio Rule (BRR)**.

Allowing t_2 to be either equal to t_0 or be set according to (9), we arrive at:

$$\phi(B) = \max_{t_2 \in \{t_0, \arg\min_{[t_0, t_0+T_e]} |BD(t)|\}} \frac{|AD(t_0)| - |BD(t_2)|}{C_{A\to B}(t_0) + \int_{t_0}^{t_2} s_B(t)dt + a(t_2 - t_0)}.$$

The motivation for considering the value $t_2 = t_0$ is that it requires no information about the future, hence it is easy to compute. Note that when evaluating the fraction for the case $B = A$ and $t_2 = t_0$ we arrive at a 0/0 indeterminate form; we arbitrarily resolve it by setting $\phi(A) = 0$. The resulting rule is a composition of greedy routing (as $t_2 = t_0$ means evaluating relays without looking into the future) and BRR, and so we term it the **Composite Rule (CR)**.

6. Evaluation

We evaluate our protocols and a number of other protocols in a **Uniform Setting** and an **Urban Setting** (based on areas of Dublin and Manhattan). The first setting, besides being very commonly used, is a good model for networks such as those comprised of Unmanned Airborne Vehicles (UAVs) [13,15]. The second is more appropriate for networks comprised of vehicles moving in cities [16].

6.1. Uniform Setting

6.1.1. Topology

We simulate 1000 nodes traveling along straight lines within a square region of side $D = 10$ km and bouncing perfectly at the boundary. Each node's initial location and direction are chosen randomly, in particular uniformly and independently of the locations and directions of the other nodes. All nodes move with speed $v_{\max} = 100$ m/s. Regarding the trajectory estimations of the nodes, each node estimates that it will maintain its current direction of travel indefinitely.

6.1.2. Transmission cost

Two models for the transmission cost are considered: under the **Quadratic Transmission Cost**, the distance cost function is $c(d) = d^2$. This model makes sense when the cost is the energy dissipated per transmission and the transmitters can perform power control. It also makes sense when we want to improve the spatial reuse of the bandwidth [17]. Alternatively, under the **Fixed Transmission Cost**, we set $c(d) = 90{,}000$ m^2. This model makes sense when we want to minimize the number of hops, or we want to minimize the transmission energy cost given a fixed amount of energy per transmission. In both cases, the maximum communication range is $R_{\max} = 600$ m.

6.1.3. Storage cost

Two models for the storage cost are considered: under the first model, there is no storage cost, whereas under the second model nodes with odd index have a zero storage cost rate and nodes with an even index have a storage cost rate equal to $10{,}000$ m^2/s. The first model is useful for evaluating the performance of our protocols in cases where storage costs are negligible. The second model is used to examine the adaptability of our protocols to environments where some nodes are preferable to others, in terms of their storage costs. The value of the storage cost rate of the even-indexed nodes is chosen so that, typically, the effects of the transmission cost and the storage cost will be comparable.

6.1.4. Data traffic

At the start of the simulation, each node has a single packet. All packets are destined to a single Base Station

(BS) that is stationary and located at the center of the square region.

6.1.5. Simulated protocols

We simulate GR, BRR, and CR, using C/D values a ranging from $a \simeq 0$ (and corresponding to a scenario when we want to minimize the cost) to $a \simeq \infty$ (and corresponding to a scenario when we want to minimize the delay). The distances entering the calculations of FPs are Euclidean. We also simulate the **Moving Vector (MoVe)** protocol [18], and **AeroRP** [15].

Under MoVe, a node A holding a packet examines the set of nodes in its neighborhood that move towards the destination (that set might contain A) and gives the packet to that node among them scheduled to pass the closest to D. If this set of nodes is empty, then the node to keep the packet among A and its neighbors is the one currently closest to D.

Under AeroRP, a node A holding a packet for a destination location D will calculate, for each neighbor B moving towards D, the following **Time To Intercept (TTI)**:

$$\mathrm{TTI} = \frac{\Delta d - R}{s_d},$$

where Δd is the distance between B and D, R is the communication radius of B, and s_d is the instantaneous relative velocity with which B is approaching D. If A moves towards D, it also calculates its own TTI. If A and all its neighbors move away from D, or if among all nodes moving towards the destination A happens to have the smallest TTI, then A keeps the packet. Otherwise, A hands over the packet to its neighbor with the smallest TTI.

MoVe and AeroRP, as defined in [18,15], do not have a parameter than can be used to trade off cost with delay. For fairness, we introduce one such parameter by permitting nodes to send packets across distances at most equal to a **Restricted Communication Range** $R' \leqslant R_{\max}$.

Observe that, with this last modification of MoVe and AeroRP, each protocol we simulate has a single, scalar, tunable parameter (either R' or a) that can be used to trade off cost with delay in a conceptually straightforward manner. That is, increasing the parameter increases the cost but decreases the delay, as verified by the simulations. We stress that the value of this parameter is kept fixed during the forwarding of each packet, i.e., it is not adjusted dynamically depending on the conditions the packet encounters.

6.1.6. Average OC/DCs and AC/DCs

We simulate 30 different network evolutions, using different seeds, each evolution lasting 1000 s. As discussed, each node creates a single packet destined for the BS. Therefore, there are 30,000 packets, and for each we calculate its OC/DC as well as its AC/DCs for all protocols. In calculating OC/DCs, we discretize time in slots of 1 s, according to Section 3, and we use Algorithm 2. In calculating the AC/DCs, we also discretize time in slots of 1 s: FPs are calculated once every second, and in the maximizations over time, such as those of (8), only integer multiples of time are considered. We then plot the averages of all 30,000 OC/DCs and all 30,000 AC/DCs of each simulated protocol. We set $T_e \simeq \infty$.

Due to our choice of $R_{\max}$, the OC/DCs and/or AC/DCs of numerous packets are infinite for values of time less than some threshold. As a single infinite curve will turn the average curve infinite, we make the following compromise: in taking averages, infinite values are ignored, but we do not plot the average curve at a time slot if more than 1% of the averaged values in that slot are actually ∞, i.e., the outage probability exceeds 1%.

6.1.7. Results

Results appear in Fig. 2. In Plots Fig. 2(a and b) the quadratic transmission cost is used, whereas in Plots Fig. 2(c and d) the fixed transmission cost is used. In Plots Fig. 2(a and c) there is no storage cost, whereas in Plots Fig. 2(b and d) the nodes with an even index have a storage cost rate equal to $10,000\ \mathrm{m}^2/\mathrm{s}$.

Overall, our protocols achieve cost/delay tradeoffs closer to the optimum than both MoVe and AeroRP. Their advantage is more pronounced in the regions of small delays, where both MoVe and AeroRP fail to deliver the packets in more than 1% of the cases, hence their average AC/DCs are not plotted.

Of our protocols, the overall best performing is GR. Note, however, that GR is the most demanding one in terms of computations and control overhead. CR is a close follower in terms of performance, even though it requires much fewer computations and control overhead. The performance of BRR is worse, as it fails to deliver a large percentage of packets in the region of small delays. Observe that in the case of the fixed transmission cost and the non-zero storage costs (Plot Fig. 2(d)) the overall performance of CR is better than the performance of GR. There is nothing fundamentally wrong with this picture: GR strives to make an informed guess about the next best hop of a packet, using a lot of information and computations, but it turns out that the information it is using is often misleading, while the subset of information used by CR turns out to be more useful on the average.

The non-zero storage cost cases differ from the zero storage cost cases in that all curves are shifted up, particularly in the regions of large delays, as these are associated with long sojourns on nodes where storage cost accrues. Note that our protocols do consider storage costs, unlike Move and AeroRP. As a result, their performance gains over MoVe and AeroRP are more pronounced for non-zero costs.

In the case of fixed transmission cost, the OC/DCs do not get as close to the delay axis for large delays as in the quadratic cost case; as a result the performance gap of all protocols with respect to the optimal is much smaller, in relative terms. This is expected: under a quadratic transmission cost, nodes postpone the packet exchange until they get quite close to each other, leading to vanishing transmission costs. This is no longer possible when all transmissions costs are the same.

6.1.8. Inaccurate trajectory estimations

Finally, we study the case where the estimations of the nodes about their future trajectories become progressively more inaccurate. In particular, in each second, each node has a probability p of deviating from its mobility model, by picking a new direction of travel, chosen randomly

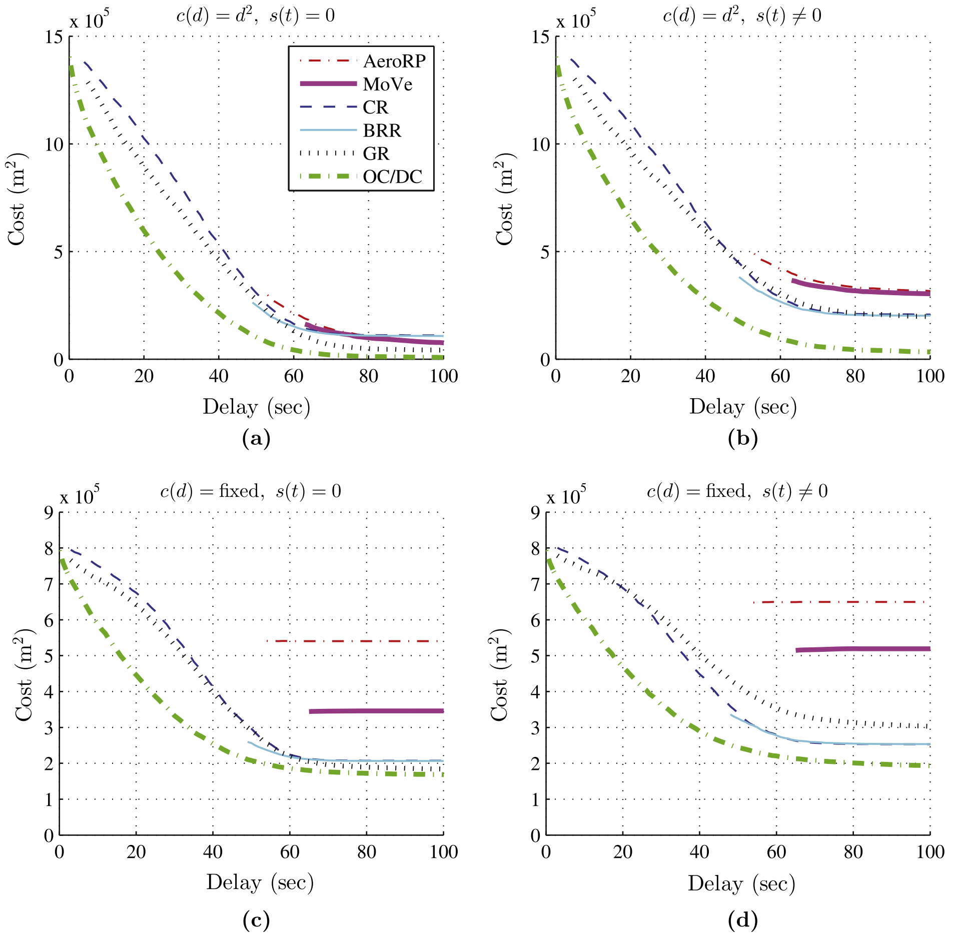

Fig. 2. AC/DC and OC/DC averages in the Uniform Setting. The legend of the first plot applies to all plots.

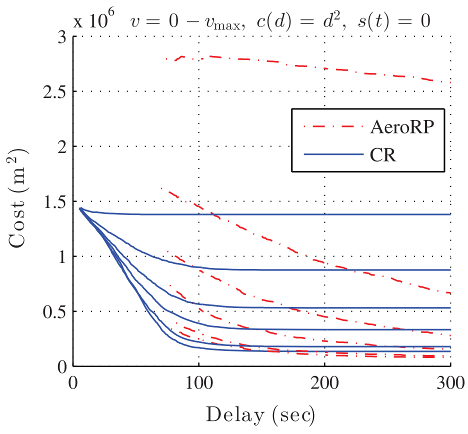

Fig. 3. AC/DC averages in the Uniform Setting for the AeroRP and CR protocols when the nodes have an imperfect knowledge of the future trajectory. The averages plotted are for $p = 0, 0.02, 0.08, 0.16, 0.32, 1$. The cost decreases as p increases.

and uniformly in $[0, 2\pi]$. In executing their routing protocol, however, nodes are oblivious to these changes, and act as if they will continue to move along the same direction indefinitely. Results for various values of p and for the AeroRP and CR protocols are plotted in Fig. 3, for node speeds selected uniformly between 0 and $v_{\max} = 100\,\text{m/s}$, for quadratic transmission costs, and no storage costs. As expected, the higher the value of p, the worse becomes the performance of both protocols, as both of them base their decisions on decreasingly relevant information. However, for low delays the performance degradation of CR is modest, as in this regime the trajectory estimates on which routing decisions are based have a short duration, and therefore are more accurate. Results for other protocols and settings were similar.

6.2. Urban setting

6.2.1. Topology

We simulate $N = 500$ nodes moving within a city, following its road network, and according to a realistic traffic mobility model, that includes interactions between vehicles, provided by the Simulation of Urban Mobility (SUMO) [19] tool. Two city environments are considered, a portion of Manhattan, and the center of Dublin, both of surface area around $16\,\text{km}^2$, and with approximately 1000 intersections each. These two cities were chosen because they complement each other well: the road network of Manhattan has a grid-like structure, whereas the road network of the center of Dublin is less structured.[3]

Regarding trajectory estimations, we assume that nodes know accurately their future trajectory up to a finite time T_e in the future, and they relay this information accurately to all their neighbors. In contrast to our Uniform Setting, we refrain from modeling imperfect trajectory estimation, as different estimation techniques might hurt each of the routing protocols we evaluate at different degrees. We do explore, however, the effects of using small values for T_e.

6.2.2. Distance metric

The distances used in the calculations of the FPs are not Euclidean. Rather, the distance $|AB|$ between two points A and B equals the distance that a vehicle will have to travel to go from A to B using the shortest path along the road network.

6.2.3. Transmission cost

A transmission across a distance d (measured in meters) has a unitless cost equal to either $c(d) = d^2$, when there is a Line of Sight (LOS) component between the transmitter and the receiver, or $c(d) = d^4$, when there is no LOS component. This is an appropriate model for the case where the cost is due to the power used by the transmitter; the transmitter power needed to reach a certain distance d increases with the distance according to an exponent which is known to be larger when there is no LOS component [20]. At distances larger than $R_{\max} = 350$ m the cost is set to infinity.

6.2.4. Data traffic

At the start of the simulation, each node possesses a single packet. All packets are destined to a single stationary Base Station (BS) located at a central location in the city.

6.2.5. Simulated protocols

We simulated GR, BRR, and CR, using C/D values a ranging from $a \simeq 0$ to $a \simeq \infty$. We did not simulate MoVe and AeroRP, as these are designed for use in settings similar to our Uniform Setting, making any comparison unfair. Rather, we simulated GeOpps [16] and D-Greedy [21], two protocols recently designed for use specifically in urban environments.

Under GeOpps, a node A that carries a packet destined for a location D calculates for itself and all its neighbors a **Minimum Estimated Time of Delivery (METD)**. This time is the sum of two terms. The first term is the time needed until the node travels to the location, along its trajectory, that is closest to D, and termed the **Nearest Point (NP)**. The second term is an estimate of the time needed for the packet to further proceed from the NP to D. In our case, we set this time equal to the time it would take the packet if it traveled from the NP to the destination with an average speed equal to the average speed of the node from its current location to the NP. As GeOpps lacks a tunable parameter that trades off cost with delay, we introduce one by imposing an upper bound $R\prime \leqslant R_{\max}$ on the maximum distance of transmissions.

Under D-Greedy, each packet starts with a **Time To Live (TTL)** counter that is reduced as time progresses. When the packet is being carried by a node A that is progressing towards its destination D with a pace that, in the hypothetical case that it could be maintained forever, would ensure that the packet is delivered in time, then the packet adopts a **Data Muling (DM)** strategy, i.e., it stays on the buffer of A. If, however, the pace of A is such that, if it was constantly maintained until the packet reached D, the packet would not make it in time, then a **Multihop Forwarding (MF)** strategy is adopted, under which the packet hops to that node among the neighbors of A that is closest to D. Of course, if node A is not moving fast enough towards D, but also is the node closest to D, the DM strategy is adopted out of necessity. In our simulations we use the TTL counter as a tunable parameter that allows the trading off between delay and cost.

6.2.6. Average OC/DCs and AC/DCs

We simulated 120 network evolutions for each city, using different seeds. As with the Uniform Setting, we discretized time in 1 s increments and we took the average of all AC/DCs and OC/DCs, ignoring infinite values, but refraining from plotting those portions of the average curves in which more than 1% of the averaged values are infinite. The averages were taken over $500 \times 120 = 60,000$ curves.

6.2.7. Results

Results are plotted in Fig. 4. Plots Fig. 4(a and b) are based on Manhattan, and Plots Fig. 4(c and d) on Dublin. For Plots Fig. 4(a and c) we assume that there are no storage costs, whereas for Plots Fig. 4(b and d) the storage cost rate is set to 0 for nodes with odd index, and equal to 10^7 s^{-1} for nodes with even index. For these plots, we assume that nodes have complete trajectory information with T_e exceeding the duration of the simulation.

With the exception of the fact that D-Greedy manages to deliver more packets (albeit at large costs) for low delays, in all cases our protocols overall perform notably better than both D-Greedy and GeOpps. This can be explained by a number of reasons. An important one is that our protocols are inherently more opportunistic, continuously scanning their nearby topology for the best it has to offer, unlike GeOpps and D-Greedy. For example, under D-Greedy, a packet that is traveling fast enough to meet its TTL deadline will never be transmitted to another node, even if one exists in the neighborhood that is traveling much faster. As another example, when GeOpps selects the node that will keep a packet, it only takes into account one metric, i.e., delay, and does not try to also keep the cost down. Furthermore, GeOpps ignores nodes that are currently moving fast towards the destination, just because they are not scheduled to pass near it. For this reason, GeOpps in particular performs worse than D-Greedy

[3] We decided against using trace sets such as those available on CRAWDAD. In all the cases of trace sets we investigated, the trace set had serious shortcomings that greatly limited its usefulness in evaluating geographic (as opposed to, say, social-based) routing; the artifacts introduced by its use, notably by ignoring the fact that vehicles interact with each other, would be more pronounced than the artifacts introduced by SUMO.

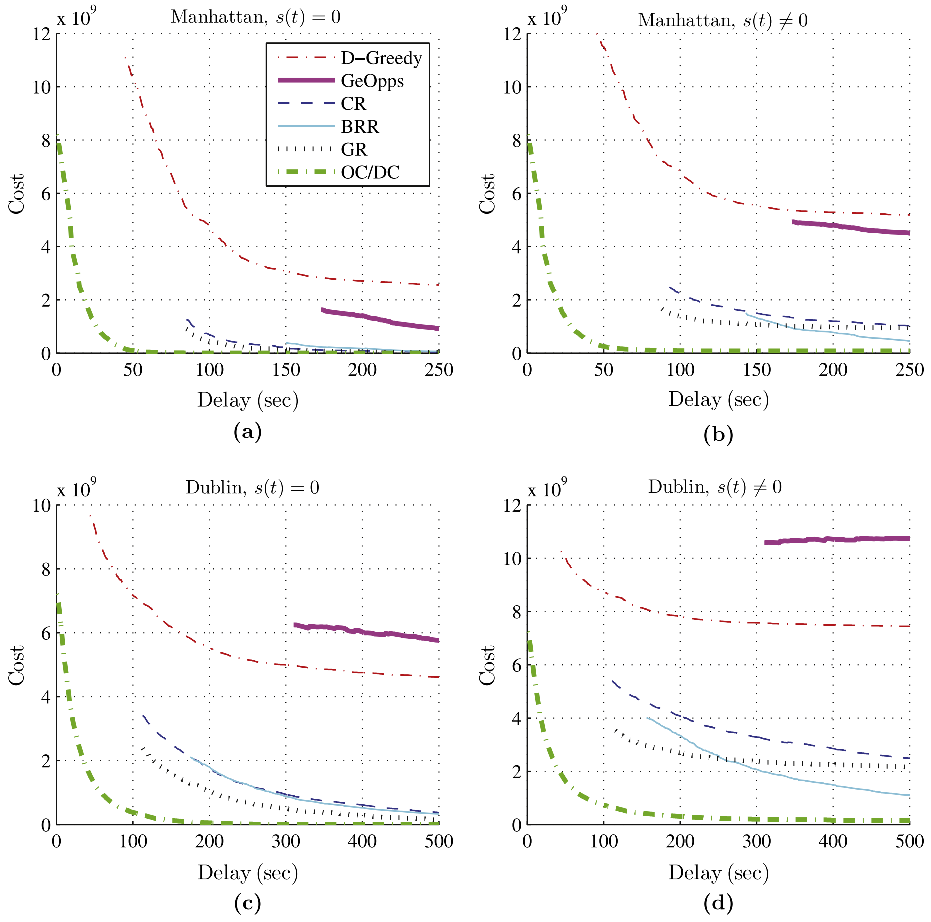

Fig. 4. AC/DC and OC/DC averages in the Urban Setting. The legend of the first plot applies to all plots.

in the low delay regime, for both cities, and also in the high delay regime of the case of Dublin, where the direction of move and the final destination are not as correlated as in Manhattan.

Observe that, as with the Uniform Setting, the gap between the performance of our protocols and that of GeOpps and D-Greedy is larger when there are storage costs. This is a direct result of the fact that GeOpps and D-Greedy do not consider storage costs. That said, the average AC/DCs of our protocols are much further away from the average OC/DC when there are such costs, which means that there is more room for improvement in this case.

6.2.8. Incomplete trajectory estimations

Finally, to evaluate the effects of having limited information about the future, we perform simulations for small values of T_e. In Fig. 5 we plot the AC/DC averages for the GeOpps and CR protocols, in the Manhattan setting with no storage costs, and for various values of T_e. Note that the control overhead needed to relay trajectory information is roughly proportional to T_e. As expected, the performance improves as T_e increases, however, for both protocols, the performance is near the optimal even for modest values of T_e. This suggests that the most useful parts of the trajectories are the early ones, at least for our setup. Results for other protocols were similar and thus are omitted.

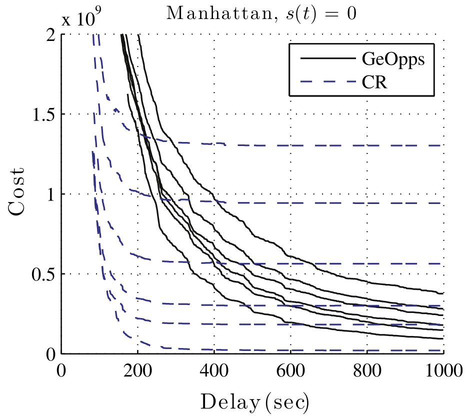

Fig. 5. AC/DC averages for the GeOpps and CR protocols when the nodes only know their trajectories during the time interval $[t_0, t_0+T_e]$, where t_0 is the present time. The averages plotted are for $T_e = 3,\ 6,\ 12,\ 25,\ 50,$ and ∞ seconds. The curves are decreasing as T increases.

7. Related work

7.1. Dynamic networks

Dynamic networks, i.e., networks that evolve with time, appear in many different application settings, and as a result have attracted the sustained interest of the Operations Research community for many years now. See, for example, the overview work [22] and references therein. The most common approach to calculating optimal flows in such networks is the one we adopt here, i.e., creating and doing calculations on a graph comprised of a cascade of subgraphs, each one related to a different epoch, called **time-expanded graph**.

A number of works have appeared that extend this line of work in the realm of DTN routing. Among the earliest are the works in [23–25], which solve a variety of shortest-path and related problems, either explicitly on DTNs, or on related communications networks. Other, more recent works on DTN problems appeared in [2,26,27].

The first part of our work, which follows this line of research, has a number of aspects that set it apart from past works. Firstly, we study a multi-objective problem (with the dual objectives of cost and delay) in terms of OC/DCs. Secondly, the time-expanded graphs we are using, i.e., the C/DEGs, have a very specific structure: the transmission links do not have any delay, and the storage links all have the same, unit delay. As a result, our C/DEG structure permits the fast, parallel calculations of the OC/DCs, using Algorithms 2 and 3. Finally, we define AC/DCs so that they are *directly comparable* to OC/DCs, and therefore bridge the gap between network optimization and the design of practicable protocols.

7.2. Tradeoffs involving delay

There are many networks, some of which not commonly studied under the DTN paradigm, where the data that much be transported are delay-tolerant, and furthermore it is possible to trade off their delivery delay with another performance metric. Handling two performance metrics jointly is a challenge, and our work provides a general framework for treating such cases. We now review some notable examples of such networks.

In the case of the Internet, when transporting delay-tolerant bulk data, on the order of Terabytes, it makes sense to take advantage of the diurnal pattern of the global traffic load and transport the data using off-peak, cheaper bandwidth. In this case, delay is exchanged with monetary cost savings [2].

Moving on to wireless networks, a common mechanism for trading off delay with other metrics is by tuning the transmitter powers. As a first example, consider the case of networks monitoring wildlife by tagging animals with sensors [3]. As the power of the transmitters attached to the animals is reduced, the batteries last longer, and hence the lifetime of the network is increased, but the network becomes more disconnected, and hence the packet delivery delay increases. As a second example, consider the case of Vehicular DTNs whose operation is supported by the addition to the network of *throwboxes*, i.e., nodes with no communication needs of their own, whose only task is to relay traffic [5]. Increasing the transmission power of a throwbox (hence decreasing its lifetime and energy efficiency, increasing its cost, and/or limiting the range of sites where it can be deployed) means that the throwbox can relay more packets, and at greater distances, while the durations of contacts between the throwbox and the other nodes in the network increase. Therefore, the packet delivery delays are decreased.

Another common mechanism encountered in wireless networks is tuning the number of copies of a packet that can exist in the network. Indeed, the more copies of a packet there are, moving with their respective holders, the faster one of them will arrive at the packet destination. However, increasing the number of copies means that more energy is dissipated (because the same packet must be transmitted multiple times), more buffer spaces are needed, and/or the aggregate end-to-end throughput is reduced (because the same packet must reserve the channel multiple times). The possible tradeoff between delay and throughput was first observed in [6], and later refined by others (see, for example, [7,8] and references therein.) The tradeoff between the delay and the energy dissipated is thoroughly investigated in [4]. There, a network connectivity model is adopted under which the intercontact times of all node pairs are exponentially distributed. The authors consider two-hop forwarding, under which only the source can create and send new packet copies to new packet relays, and Epidemic Routing, under which any node holding a packet copy can create a new copy and send it to a node it is in contact with. The authors define the problem of maximizing the packet delivery probability by a time T subject to a constraint on the average energy dissipated in the transmission of all copies. The maximization is with respect to the function $p(\cdot)$ where $p(t)$ is the probability that an eligible node will create a packet copy upon its contact with an eligible packet relay at time t, given that the packet was created at time instant $t = 0$.

7.3. Applying the OC/DC, AC/DC formulation

Our formulation of Sections 2–4 can help the designers of networks such as those mentioned in Section 7.2 study the discussed tradeoffs. The overall method, which we also applied in Sections 5 and 6, is the following: First, the designer specifies the network model, including restrictions (such as the maximum transmission range) that must hold under any routing protocol. Based on these, the designer calculates the average OC/DC. Then, the designer formulates a specific protocol, where the tunable parameters comprise the parameter vector. The designer then calculates, either by analysis or simulation, the resulting average AC/DC, and compares it to the average OC/DC.

As a parting example of this process, in Fig. 6 we plot average AC/DCs for the Uniform Setting and for three protocols that are based on packet replication: Epidemic Routing, Spray & Wait, and Binary Spray & Wait [28]. The

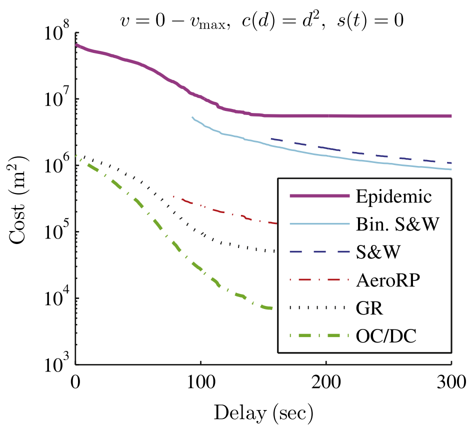

Fig. 6. AC/DC averages in the Uniform Setting for Epidemic Routing, Binary Spray & Wait, Spray & Wait, AeroRP, and the General Rule. The OC/DC averages are also plotted. Note that the y-axis scale is logarithmic.

tunable parameters are the transmission radius, for the case of Epidemic Routing, and the number of copies, for the cases of Spray & Wait and Binary Spray & Wait. For reasons of comparison, we also plot the average AC/DCs of AeroRP and GR, and the average OC/DC. We assume the quadratic transmission cost model, no storage costs, and node speeds uniformly distributed in the range from 0 m/s to $v_{\max} = 100$ m/s. As expected, the protocols making use of geographic information perform orders of magnitude better than the protocols that do not have such information available, but the comparison is clearly not fair.

Using this process comes with limitations. Notably, the information on how to achieve a particular point on an OC/DC or AC/DC is not encapsulated in that curve. Also, single AC/DCs and OC/DCs are only pertinent to a particular network topology and source–destination pair, hence we cannot draw conclusions on a protocol's performance from a single curve. We alleviate this problem by taking averages of many curves over many randomly created topologies. This averaging is similar to the averaging of many specific packet delays in order to calculate the average delay of a protocol. However, we average whole curves instead of single numbers. A drastically different approach, that dispenses with these limitations, is to adopt a stochastic setting and calculate stochastic quantities. See for example [4,29,30,17,31]. The limitation of this approach is that, in order to arrive at analytical results, it is necessary to use relatively simple network models, which are not tightly related to real-life scenarios.

7.4. Node selfishness

Node selfishness is a major challenge in the design of networks of autonomous nodes. In the case of wireless DTNs, selfish nodes may, or will certainly, refuse to relay the other nodes' packets. There are many reasons for this: nodes are often severely constrained in their resources and hence motivated to be selfish, typically they belong to disparate users who are not motivated to help each other, and, finally, due to the large delays involved and the disconnected nature of the network, it is difficult for each node to monitor and evaluate the behavior of others.

As a result, a number of works have recently focused on the issue of node selfishness in wireless DTNs. One of the earliest such works is [32]. There, various models of selfishness are introduced, and their effects on the performance of routing is simulated using real-life traces and social network models. It is found that, in all cases, the network is robust with respect to the type of selfish behavior, due to the existence of multiple forwarding paths for each packet.

In [33], nodes are divided in two classes: selfish and not selfish. Nodes that are not selfish always receive a packet destined for another node, and further transmit it, provided these actions are specified by the protocol used. Selfish nodes, on the other hand, refrain from copying or further forwarding such a packet with probabilities p_{nc} and p_{nf} respectively. The propagation of a packet in the network is modeled as a two dimensional Continuous Time Markov Chain (CTMC), where the non-absorbing state $(n(t), k(t))_{t\geqslant 0}$ signifies that $n(t)$ nodes have a packet copy, of which $k(t)$ are selfish. State transitions are calculated assuming exponential intercontact times, and the performance of the routing protocol is evaluated in terms of the time it takes the CTMC to arrive at the absorbing state where the packet has arrived to its destination.

In [30] the authors adopt a CTMC setting in order to evaluate the performance of a wireless DTN when nodes exhibit *social selfishness*. Social selfishness in an important concept that captures the fact that nodes are more interested in behaving altruistically towards their peers and members of the same community than towards other nodes. In this case, nodes receive and forward other nodes' packet copies depending on whether they belong to the same out of two communities. As in our work, two performance metrics are used, the message delivery delay, defined as the average time it takes for the CTMC to reach an absorbing state where the destination has the packet, and the message delivery cost, defined as the average number of node replicas circulating at the time of the delivery of a packet copy to a destination. In [29] this setting is extended in the context of DTN multicasting, i.e., when the created packet must be delivered to multiple nodes.

The framework we have developed does not explicitly model selfishness. However, selfish behavior can be incorporated in the protocol model, and so is compatible with our framework. For example, the level of selfishness of a node, and the levels of selfishness of the nodes it is expected to meet, could be taken into account in the calculation of the forwarding potential (8) of Section 5.

Furthermore, it is possible to study selfishness with our formulation by modeling selfish behavior through the parameter vector **p**. In this case, the derived AC/DCs capture the tradeoff that emerges between the packet delivery delay and the communication cost as the selfishness varies. Indeed, we expect that increasing node selfishness will increase the packet delay but decrease the communication cost. In contrast to the previously discussed cases, the parameter vector will not be not tunable by the designer, unless incentives are provided, in which case *our formulation can help in the design of the incentive mechanisms.*

8. Conclusions

Previous research on DTNs has studied tradeoffs between the packet delivery delay and specific other metrics [2–8]. In the first part of this work, we take a novel, more general approach: we develop a framework under which the tradeoff between the packet delivery delay and an abstract transportation cost is quantified, in terms of the OC/DC when nodes route optimally, and in terms of the AC/DC when nodes use a suboptimal routing protocol.

Our framework can be applied to a variety of settings. As an example, in the second part of this work, we focus on the case of mobile wireless DTNs. We develop a family of hybrid delay-tolerant/geographic routing protocols, and we show that their average AC/DCs lie much closer to the average OC/DC than the average AC/DCs of other, recently proposed protocols. As expected, the more information is available to a protocol, the closer its average AC/DC follows the average OC/DC.

Future research includes the application of the first part of this work to the study of node selfishness, as discussed in Section 7.4.

Acknowledgements

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program ‘‘Education and Lifelong Learning’’ of the National Strategic Reference Framework (NSRF) Research Funding Program: THALES. Investing in knowledge society through the European Social Fund.

The authors wish to thank Dr. Savvas Gitzenis as well as the anonymous reviewers for their numerous insightful comments that greatly improved the exposition.

References

[1] A.G. Tasiopoulos, C. Tsiaras, S. Toumpis, On the cost/delay tradeoff of wireless delay tolerant geographic routing, in: Proc. IEEE WOWMOM, San Francisco, CA, June 2012.
[2] N. Laoutaris, G. Smaragdakis, P. Rodriguez, R. Sundaram, Delay tolerant bulk data transfers on the internet, in: Proc. ACM SIGMETRICS, Seattle, WA, June 2009, pp. 229–238.
[3] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, D. Rubenstein, Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet, in: Proc. ASPLOS-X, San Jose, CA, October 2002, pp. 96–107.
[4] Y. Li, Y. Jiang, D. Jin, L. Su, L. Zeng, D.O. Wu, Energy-efficient optimal opportunistic forwarding for delay-tolerant networks, IEEE Trans. Veh. Technol. 59 (9) (2010) 4500–4512.
[5] N. Banerjee, M.D. Corner, B.N. Levine, Design and field experimentation of an energy-efficient architecture for DTN throwboxes, IEEE/ACM Trans. Network. 18 (2) (2010) 554–567.
[6] M. Grossglauser, D.N.C. Tse, Mobility increases the capacity of ad hoc wireless networks, IEEE/ACM Trans. Network. 10 (4) (2002) 477–486.
[7] S. Toumpis, A.J. Goldsmith, Large wireless networks under fading, mobility, and delay constraints, in: Proc. IEEE INFOCOM, Hong Kong, China, March 2004.
[8] G. Sharma, R. Mazumdar, N. Shroff, Delay and capacity trade-offs in mobile ad hoc networks: a global perspective, IEEE/ACM Trans. Network. 15 (5) (2007) 981–992.
[9] G. Papastergiou, I. Psaras, V. Tsaoussidis, Deep-space transport protocol: a novel transport scheme for space DTNs, Comput. Commun. 32 (16) (2009) 1757–1767.
[10] J. Burgess, B. Gallagher, D. Jensen, B.N. Levine, MaxProp: routing for vehicle-based disruption-tolerant networks, in: Proc. IEEE INFOCOM, Barcelona, Spain, April 2006, pp. 1–11.
[11] A. Ferreira, A. Goldman, J. Monteiro, Performance evaluation of routing protocols for MANETs with known connectivity patterns using evolving graphs, Wirel. Netw. 16 (3) (2010) 627–640.
[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, third ed., MIT Press, 2009.
[13] E. Kuiper, S. Nadjm-Tehrani, Geographical routing with location service in intermittently connected MANETs, IEEE Trans. Veh. Technol. 60 (2) (2011) 592–604.
[14] R. Catanuto, S. Toumpis, G. Morabito, On asymptotically optimal routing in large wireless networks and geometrical optics analogy, Comput. Netw. 53 (11) (2009) 1939–1955.
[15] K. Peters, A. Jabbar, E.K. Çetinkaya, J.P.G. Sterbenz, A geographical routing protocol for highly-dynamic aeronautical networks, in: Proc. IEEE WCNC, Cancun, Mexico, March 2011, pp. 492–497.
[16] I. Leontiadis, C. Mascolo, GeOpps: geographical opportunistic routing for vehicular networks, in: Proc. IEEE WOWMOM, Espoo, Finland, June 2007, pp. 1–6.
[17] A. Sidera, S. Toumpis, Delay tolerant firework routing: a geographical routing protocol for wireless delay tolerant networks, EURASIP J. Wirel. Commun. Network. (2013).
[18] J. LeBrun, C.-N. Chuah, D. Ghosal, M. Zhang, Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks, in: Proc. IEEE VTC Spring, vol. 4, Stockholm, Sweden, May–June 2005, pp. 2289–2293.
[19] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, SUMO – Simulation of Urban MObility: an overview, in: Proc. SIMUL, Barcelona, Spain, October 2011, pp. 55–60.
[20] A. Goldsmith, Wireless Communications, Cambridge University Press, 2005.
[21] A. Skordylis, N. Trigoni, Efficient data propagation in traffic-monitoring vehicular networks, IEEE Trans. Intell. Transport. Syst. 12 (3) (2011) 680–694.
[22] M. Skutella, An introduction to network flows over time, in: Research Trends in Combinatorial Optimization, 2009, pp. 451–482.
[23] A. Ferreira, On models and algorithms for dynamic communication networks: the case for evolving graphs, in: Proc. ALGOTEL, Mèze, France, May 2002, pp. 155–161.
[24] S. Merugu, M. Ammar, E. Zegura, Routing in space and time in networks with predictable mobility, Georgia Institute of Technology, Tech. Rep. GIT-CC-04-07, 2004.
[25] S. Jain, K. Fall, R. Patra, Routing in a delay tolerant network, in: Proc. ACM SIGCOMM, Portland, OR, August–September 2004, pp. 145–158.
[26] D. Hay, P. Giaccone, Optimal routing and scheduling for deterministic delay tolerant networks, in: Proc. WONS, Snowbird, UT, February 2009, pp. 27–34.
[27] G. Konidaris, S. Toumpis, S. Gitzenis, Primal decomposition and online algorithms for flow optimization in wireless DTNs, in: Proc. IEEE GLOBECOM, Atlanta, GA, December 2013.
[28] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Efficient routing in intermittently connected mobile networks: the multiple-copy case, IEEE/ACM Trans. Netw. 16 (1) (2008) 77–90.
[29] Y. Li, G. Su, D.O. Wu, D. Jin, L. Su, L. Zeng, The impact of node selfishness on multicasting in delay tolerant networks, IEEE Trans. Veh. Technol. 60 (5) (2011) 2224–2238.
[30] Y. Li, P. Hui, D. Jin, L. Su, L. Zeng, Evaluating the impact of social selfishness on the epidemic routing in delay tolerant networks, IEEE Commun. Lett. 14 (11) (2010) 1026–1028.
[31] A. Sidera, S. Toumpis, On the delay/cost tradeoff in wireless mobile delay-tolerant networks, in: Proc. Wiopt, Hammamet, Tunisia, May 2014.
[32] P. Hui, K. Xu, V.O.K. Li, J. Crowcroft, V. Latora, P. Lio, Selfishness, altruism and message spreading in mobile social networks, in: Proc. IEEE NetSciCom, Rio de Janeiro, Brazil, April 2009.
[33] M. Karaliopoulos, Assessing the vulnerability of DTN data relaying schemes to node selfishness, IEEE Commun. Lett. 13 (12) (2009) 923–925.

Argyrios G. Tasiopoulos received the B.Sc. in Informatics and the M.Sc. in Computer Science, by the Department of Informatics of the Athens University of Economics and Business, in 2010 and 2012 respectively. In 2013 he was employed as a researcher by the NSRF research program "DISCO", through the Program "Education and Lifelong Learning 2007–2013". He is currently a Ph.D. student at the Department of Electronic and Electrical Engineering, University College London. His research interests include network optimization and algorithms, and their application to wireless networks from an information-centric perspective.

Christos Tsiaras is a member of the Communication Systems Group of the Department of Informatics (IFI) of the University of Zurich (UZH), Switzerland. He holds a M.Sc. in Computer Science, with a specialization in Computer and Communication Networks, from the Department of Informatics of the Athens University of Economics and Business (AUEB) in Greece and a B.Sc. in Physics from the University of Crete (UoC) in Greece. His main research interests are wireless and mobile communications, protocols design, accounting and charging, and cloud networking.

Stavros Toumpis received the Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece, in 1997, the M.S. degrees in Electrical Engineering and Mathematics from Stanford University, CA, in 1999 and 2002, respectively, and the Ph.D. degree in Electrical Engineering, also from Stanford, in 2003. He is currently an Assistant Professor at the Department of Informatics of the Athens University of Economics and Business, Athens, Greece. His research is on wireless ad hoc networks, with emphasis on capacity, network optimization, and information theoretic issues.