Dynamic Resource Management in SDN-based Virtualized Networks

Rashid Mijumbi*, Joan Serrat*, Javier Rubio-Loyola†, Niels Bouten‡, Filip De Turck‡ and Steven Latré§
*Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
†CINVESTAV, 87130 Ciudad Victoria, Tamaulipas, Mexico
‡Ghent University – iMinds, B-9050 Gent, Belgium
§University of Antwerp – iMinds, B-2020 Antwerp, Belgium

***Abstract*—Network virtualization allows for an abstraction between user and physical resources by letting a given physical infrastructure to be shared by multiple service providers. However, network virtualization presents some challenges, such as, efficient resource management, fast provisioning and scalability. By separating a network's control logic from the underlying routers and switches, software defined networking (SDN) promises an unprecedented simplification in network programmability, management and innovation by service providers, and hence, its control model presents itself as a candidate solution to the challenges in network virtualization. In this paper, we use the SDN control plane to efficiently manage resources in virtualized networks by dynamically adjusting the virtual network (VN) to substrate network (SN) mappings based on network status. We extend an SDN controller to monitor the resource utilisation of VNs, as well as the average loading of SN links and switches, and use this information to proactively add or remove flow rules from the switches. Simulations show that, compared with three state-of-art approaches, our proposal improves the VN acceptance ratio by about 40% and reduces VN resource costs by over 10%.**

***Keywords*—Future Internet, network virtualization, software defined networking, VN mapping, dynamic resource management.**

I. Introduction

Network virtualization has emerged as a promising technology for the future Internet in which network deployment and management are separated from service provision [1]. Specifically, an infrastructure provider (InP) owns, controls and manages physical resources in form of substrate networks (SNs), which may be used by one or more service providers (SPs) to create virtual networks (VNs) to provide services to end-users. However, hosting multiple VNs and supporting their complete isolation raises resource management (RM) challenges for the InP, e.g. the need to *efficiently* alocate SN resources to multiple VNs.

Software Defined Networking (SDN) [2] is an appealing platform for network virtualization environments (NVE), since each VN's control logic can run on a controller rather than the physical switches [3]. SDN allows for a flexible and easier way of defining VNs, say, by representing each virtual link as a flow and hence defining a VN as a set of flow rules in different switches. This way, SDN's control plane can be used to achieve important resource allocation policies such as SN load balancing, VN resource cost minimization, e.t.c. For instance, Flowvisor [4] and XNetMon [5] allow multiple tenants to share an SDN substrate through virtualization by allowing for isolation and sharing of network slices.

However, current proposals for virtualized SDNs are silent about the RM requirements that result in such an environment. For example, an important step in initializing VNs is the mapping of virtual nodes[1] and links to substrate nodes and links. While this mapping is a well studied problem [6], as we show in this paper, some of the resource mapping approaches such as path splitting [7] that have been shown to lead to better resource utilisation in VNs create another problem in an SDN environment. When a virtual flow is split into multiple sub-flows, each sub-flow would need flow rules in each of the switches along the substrate path that supports it, hence requiring more ternary content-addressable memory (TCAM), which is expensive to build, consumes a lot of power and dissipates a high level of heat [8]. In addition, if performed in a static way, virtual to substrate resource mapping leads to high resource fragmentation at the SN layer [9]. Therefore, since VN requests arrive and depart in a dynamic manner, dynamic RM leads to better resource utilisation efficiency [10]. Current approaches to dynamic RM in virtualized networks are mainly based on link migration [6], which is aimed at balancing the load on substrate links without considering the effect on the substrate node resources. As already mentioned, given the cost and power dissipation [11], [12] of node resources in SDN environments, it is necessary for a RM approach to also be node resource aware and manage them.

In this paper, we propose a flow migration approach to dynamically manage link and switch resources in an SDN-based virtualized environment which does not only consider link resources, but also node resources and VN resource costs. To this end, we extend a floodlight controller [13] by adding an application module which monitors the resource costs of mapped virtual links, as well as average load of the substrate links and switches. This information, coupled with that about arrivals and departures of VN requests is used to determine which virtual links can profitably be migrated. The module then proactively modifies (adds and/or deletes) flow rules (which represent virtual links) from the affected switches. The idea of our proposal is that due to the dynamic arrival of VN requests, some virtual flows may utilise more resources at the time of mapping, but when some VNs leave, more efficient flows can be established.

The rest of the paper is organised as follows: We present related work in Section II. Section III describes the problem for which we make a proposal in Section IV. Our proposal is evaluated and discussed in Section V, and the paper concluded in Section VI.

[1] In this paper, the terms *node* and *switch* are used synonymously.

II. Related Work

While SDN itself does not directly virtualize a network, it has already received attention with regard to virtualization, with [3], [4], all proposing closely related abstraction layer approaches based on OpenFlow to enable the sharing of the same hardware data plane among multiple logical networks. These proposals do not consider the management of link and switch resources. [14] presents an offline and static mixed integer linear programming (MILP) formulation for a centralised controller to calculate optimal end-to-end virtual paths over the underlying InP, considering multiple requests simultaneously. The authors in [15] propose a VN mapping approach in SDN that aims at balancing the load on the SN and minimizing controller-to-switch delays. They perform VN mappings in an offline manner, assuming all VN requests are known in advance. [12] proposes integrated allocation of link bandwidth and flow table for multiple control applications in SDN using a price-based joint allocation model of network resource in order to achieve the minimum global delay.

With regard to dynamic RM in virtualized networks, Reactive VN resource management approaches are given in [16] in which a VN reconfiguration scheme migrates a single node (and its associated VN links) in order to control migration costs, and in [9] where a VN reconfiguration solution for optical networks performs multiple (batch) reconfigurations. Both these approaches are reactive in a way that migrations are only performed after, say, a failed mapping. They also do not consider switch loading. The proposal in [7] performs a selective VN reconfiguration scheme that prioritizes the reconfiguration for overloaded VNs and migrates them, while [17] employs path migration to periodically re-optimize the utilization of the SN by recomputing VN links for active VN requests with longer residual lifetimes. Both approaches do migrate the whole VN at once, employ path splitting (which leads to more switch resource utilization), and the migrations do not take switch loading into consideration. Finally, [10] proposes an approach for dynamically managing VN resources, but it does not not involve virtual link migrations.

In summary, current proposals for a SDN-based VNs are silent about the RM requirements of NVEs, while the approaches that are not geared towards SDN may become inefficient in such environments, especially because they do not take into consideration the need to also manage switch resources. Our approach considers not only substrate link loading, but also switch loading and the cost of mapping VNs. Giving consideration to switch resources is necessary in an SDN environment given that switch memory is more expensive. To the best of our knowledge, this is the first attempt to dynamically manage resources in SDN based NVEs.

III. Problem Description

The RM problem considered in this paper is shown by Fig. 1. VN requests arrive one at a time to the SN and a mapping is performed. If the mapping is successful, the dynamic RM module is triggered, otherwise, other VN requests are considered. The focus of this paper is on the dynamic RM block which ensures that even after a given VN is mapped (during VN service period), link remappings/migrations can be done to ensure efficient resource utilisation. It also ensures that after the VN departs, the remaining VNs can be migrated to optimise resource usage.

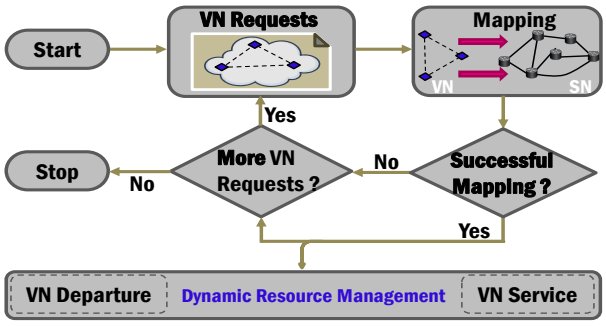

Fig. 1. Resource Management Algorithm

A. Network Representation

We model substrate SDN as a weighted undirected graph, $G(S, L)$, where S and L represent the sets of substrate switches and links, respectively. Each substrate link $l_{uv} \in L$ connecting the substrate switches u and v has a bandwidth B_{uv}, while each substrate switch $u \in S$ has a memory M_u. The memory is the switch's TCAM [18], and is therefore a measure of the number of flow entries that the switch can contain. In the same way, we represent a VN slice as a weighted undirected graph denoted by $G'(S', L')$, where S' and L' represent the sets of virtual switches and links respectively. Each virtual switch $u' \in S'$ has a memory $M'_{u'}$, while each virtual link $l_{u'v'} \in L'$ connecting the virtual switches u' and v' has bandwidth $B'_{u'v'}$

B. Virtual to Substrate Network Mapping

An example of a mapping problem is shown in the lower part of Fig. 2 where the virtual network ABC is being mapped onto the substrate network PQRSTU. To map the VN onto the SN, each virtual switch $i \in (A, B, C)$ should be mapped onto a unique substrate switch $u \in (P, Q, R, S, T, U)$ that has enough memory to support it. In addition, all the virtual links have to be mapped to one or more substrate links connecting the switches to which the virtual switches at its ends have been mapped. Each of the substrate links must have a sufficient bandwidth to support the virtual link. This mapping step can be achieved by use of most state-of-the-art mapping algorithms [6], and is therefore out of the scope of this paper.

However, the resulting mapping in Fig. 2 exposes a problem that is unique to an SDN network. Consider the virtual links AB and AC, which are mapped onto substrate paths PTQ and PTU respectively, and hence, both go through the substrate switch T. At the VN layer, flow rules would only be required in virtual switch A to direct traffic to switches B and C respectively, yet at the physical network layer, we would need extra rules for each of the flows in switch T to ensure that the two flows end up in the intended switches. In fact, the number of rules would even increase further in case of path splitting. For example, if the flow AC splits at T to take the paths TSU and TU, then another rule would be needed in switch S. Having a high number of rules in the network for a single flow does not only mean that VN owners have to pay more for their network slices (increased costs), but also ensures that SN switch resources (memory) are easily depleted, which causes subsequent VN requests to be rejected. This negatively affects the VN acceptance ratio and

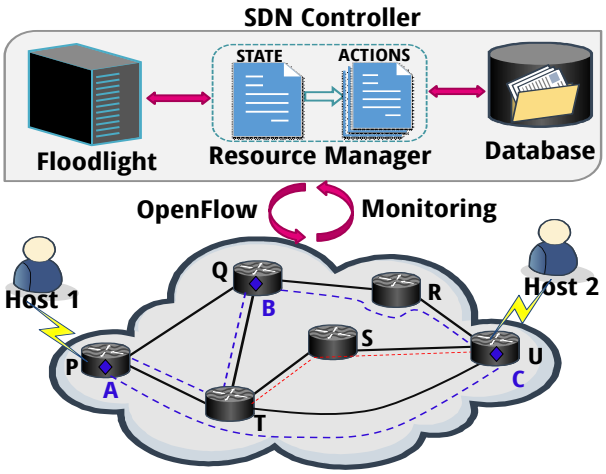

Fig. 2. Proposed Dynamic Resource Management System

hence profitability of InPs. Therefore, there is a need for an efficient management of resources (especially the memory of the switches along the substrate path) to ensure that in addition to link bandwidth which has been the focus of current path migration approaches, switch memory is also considered.

C. Dynamic Resource Management

The dynamic RM proposed in this paper takes effect after a successful mapping. It involves monitoring the resource utilisation (both link and switch) of mapped virtual links, as well as average loading of the substrate links and switches. Every time substrate resources are freed (because of departures of VNs), remapping attempts are made for those virtual links that are deemed to be more deserving of a remapping. The objectives of the remappings are: balance SN loading and minimize the total resources used by the VN. Our proposal is detailed in the next Section.

IV. Proposed Resource Management System

The proposed RM approach is shown in Fig. 2. Compared to the general SDN network, our system includes a resource manager and a database, which are extensions to an existing SDN controller, floodlight. The forwarding module in floodlight (which is responsible for automatically and reactively managing flow rules) is disabled such that all flow rules in the switches are added proactively by the resource manager, which allows better manageability.

Database

The database is used to store all flow rules that are currently active in the SN. The idea is that instead of continuously polling the switches to establish their tables over the southbound, the controller maintains an entry of all the active rules. Every time a VN mapping is successful[2], all the flow rules that represent the mapped virtual links are added to the database, and at the same time they are pushed to the affected switches using OpenFlow. The rules are added with practically no time out to ensure that they are only removed from the switches by the resource manager. In the same way, when a rule is deleted from any of the switches, it is also deleted from the database. The database also stores the virtual to substrate link and switch mappings, as well as the resource availability and usage statistics.

[2]A VN to SN mapping algorithm is implemented as part of the resource manager

Resource Manager

The resource manager is state-action engine, that, based on a specific *state* takes an *action*.

A. State

The states are determined from the dynamic nature of the virtualization environment. The network state changes as new VNs are embedded and as old ones leave. These changes affect the loading of the SN as well as the resource utilisation of the VNs. These parameters are defined for each virtual link, and are aimed at determining a measure of the desirability of migrating a given virtual link. The idea is to re-allocate those virtual links that maximise the benefit, i.e reduce substrate resource utilisation, while at the same time reducing the load of the already loaded substrate links and switches.

1) Virtual Link Resource Utilisation: We define the resource utilisation $R_{u'v'}$ of a virtual link $l_{u'v'}$ as the total resources used by the virtual link. It takes into consideration both the bandwidth $B_{u'v'}$ used by the virtual flows of the virtual link on each substrate link onto which it is mapped, as well as the switch memory $M_{u'}$ used by the flow rules associated with it, both at the end switches u' and v' as well as intermediate ones. Whenever any VN is mapped onto the SN, each of its virtual links is evaluated to determine $R_{u'v'}$ using the expression in (1).

$$R_{u'v'} = \alpha \sum_{l_{uv} \in P_{l_{u'v'}}} \left(B_{u'v'}\right) + \beta \sum_{u \in P_{l_{u'v'}}} \left(M_{u'}\right) \quad (1)$$

where $P_{l_{u'v'}}$ is the substrate path onto which the virtual link $l_{u'v'}$ has been mapped. As an example, using the networks in Fig. 2, the substrate path P_{AB} for virtual link AB is PTQ. Therefore, the set $l_{uv} \in P_{l_{u'v'}}$ includes both PT and TQ, while $u \in P_{l_{u'v'}}$ includes the switches P, T, and Q. α and β are constants aimed at scaling the two terms to comparable magnitudes, or for giving one resource type more importance than the other.

2) Available Substrate Resources: The available substrate resources $A_{u'v'}$ is also defined for each virtual link $l_{u'v'}$. It is defined as a sum over all links and switches on the substrate path $P_{l_{u'v'}}$. It is the total amount of both substrate link and switch resources that have not been allocated (available), i.e. total capacity *minus* resources allocated to mapped VNs. The objective of using available resources is to ensure that substrate links or switches with less available resources are given priority when decisions about which virtual links to migrate are made. $A_{u'v'}$ is defined in (2) below.

$$A_{u'v'} = \alpha \sum_{l_{uv} \in P_{l_{u'v'}}} \left(B_{uv} - B'_{uv}\right) + \beta \sum_{u \in P_{l_{u'v'}}} \left(M_u - M'_u\right) \quad (2)$$

where B'_{uv} is the total bandwidth of substrate link l_{uv} that is currently allocated to virtual links, while M'_u is the total memory of substrate switch u that is currently allocated to the virtual switches mapped onto it.

3) Virtual Link Weights: Finally, with the two variables defined in sub sections IV-A1 and IV-A2, we determine a weight $W_{u'v'}$ which is attached to each virtual link. This weight is a linear combination of equations (1) and (2), and is given in (3).

$$W_{u'v'} = \lambda R_{u'v'} - \mu A_{u'v'} \quad (3)$$

where λ and μ are constants aimed at biasing the weight of a given virtual link to either its resource utilisation or the availability of resources on the substrate links onto which it is mapped. A high value of $W_{u'v'}$ means that the virtual link $l_{u'v'}$ utilizes a lot of SN resources (i.e. $l_{u'v'}$ is not efficiently mapped) and that the SN is highly loaded along the path mapping $l_{u'v'}$. This in turn represents a virtual link which should be urgently considered for a remapping. A similar (opposite) argument holds for low values of $W_{u'v'}$. For a given virtual link, the values of $R_{u'v'}$ and $A_{u'v'}$ are uncorrelated, and may be conflicting in some cases since the substrate path which results into lower resource utilisation may have less available resources. The values of λ and μ may also be used to scale the values of $R_{u'v'}$ and $A_{u'v'}$ to comparable magnitudes[3]. The reason for the negative sign on the term $A_{u'v'}$ is that, as earlier stated, we are interested in favouring those substrate links or switches which have the least available resources.

4) Flow Migration: Every time the status of the SN changes (i.e a new VN request is accepted or a previously accepted request departs), the resource manager re-computes the weight $W_{u'v'}$ for each virtual link $l_{u'v'}$. The database includes a map $V_w(l_{u'v'}, W_{u'v'})$ of these weights, which is sorted in descending weight every time it is changed. If the status change was due to a departing VN, a number n of virtual links is selected from the top of V_w, and for all of them, starting with the top most one a re-mapping is attempted. In this paper, n is chosen to be equal to the number of links in the departing VN. This choice of n is reasonable because since resources from n substrate paths are freed by the departing VN, which also avoids the computation costs that would result from frequently attempting to remap all the virtual links.

If the re-mapping attempt is successful, a new weight $W^{new}_{u'v'}$ is calculated for the link, and compared with the old one. A given virtual link is only actually migrated if its new weight is less than the previous one. This avoids migrating a virtual link into a worse mapping, since it is possible that the resources freed by the departing VN do not result into a better substrate path of the virtual link under consideration.

B. Actions

The actions are with respect to modification of rules in the switches. Therefore, there are three possible actions: addition, modification and deletion of flow rules from the switches. All these actions are supported by Openflow.

1) Flow Rule Addition: Whenever a new VN is successfully mapped, the resource manager, through floodlight, adds flow rules to the switches concerned, for all the virtual links. For example, after the mapping of the virtual network ABC in Fig. 2, for virtual link AB, the controller would add rules to switches P, T and Q, to establish the substrate path PTQ. These three rules are also added to the database, and the resource availability and utilisation updated.

2) Flow Rule Modification: As VNs arrive and depart, the SN becomes fragmented. It is possible that after sometime, the flow rules for a virtual link should be modified. This is triggered by a departure of a VN which avails some resources in a better substrate path. For instance, if resources become available on the substrate link PQ, it may be better to migrate the virtual link AB from PTQ to PQ. In this case, the rule in switch P should be modified to end up on Q instead of T.

3) Flow Rule Deletion: The resource manager can delete a flow rule in two cases: (1) when a virtual link is migrated, some rules become useless and should therefore be deleted. As an example, in Fig. 2, if the virtual link AB is migrated from substrate path PTQ to PQ, then the corresponding flow rule in switch T becomes useless, and is therefore deleted, (2) When a given virtual link departs, all its links are taken down, and as such all its corresponding rules are deleted.

V. Performance Evaluation

A. Simulation Setup

To evaluate our proposal, we extended a floodlight controller to include a module that performs the functions of our proposal. The SN is created in Mininet [19]. Both Floodlight and Mininet run on different Ubuntu virtual machines each with 1.0GB RAM. The SN topology is based on the GÉANT [20] network. GÉANT is composed of 23 routers interconnected using 38 links. The VN topologies are created using Brite [21] with a uniformly distributed number of nodes between 3 and 10. The virtual to SN mapping is performed by performing the node mapping using a greedy approach [7] and link mapping using a multi-commodity flow (MCF) [22] formulation (without path splitting), which is then solved using CPLEX 12.6.0.0 [23]. The memory and bandwidth capacities of substrate switches and links are uniformly distributed between 100 and 250 units respectively. The memory demand for virtual network switches is uniformly distributed between 2 and 10 units, while the bandwidth demand of the links is uniformly distributed between 25 and 50 units. We consider that each flow rule requires 1 unit of switch memory. We assume Poisson arrivals at an average rate of 1 per 5 time units. The average service time of the requests is 120 time units and assumed to follow a negative exponential distribution. The simulation is performed for 1500 VN arrivals.

B. Evaluation Parameters

1) Acceptance Ratio: The acceptance ratio is a measure of the number of VN requests accepted compared to the total requests. In the long run, the acceptance ratio translates into profitability of infrastructure providers, since each accepted VN request would result into revenue for the InP.

2) Average Link and Switch Resource Utilization: We define average link resource utilization as the average proportion of the total substrate link bandwidth capacity that is under use at any given time. In the same way, we define the

[3]In this paper, we use them for this latter purpose

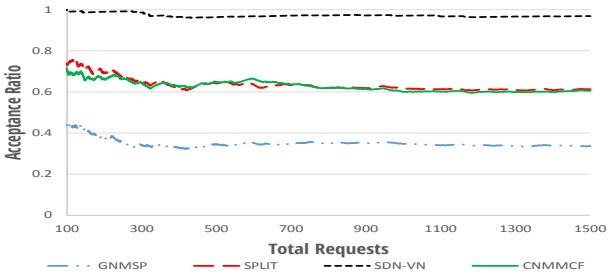

Fig. 3. Average Acceptance Ratio

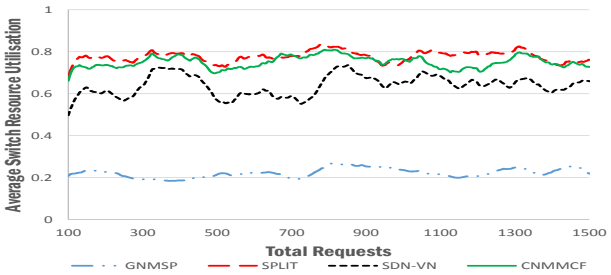

Fig. 4. Average Switch Resources Utilization

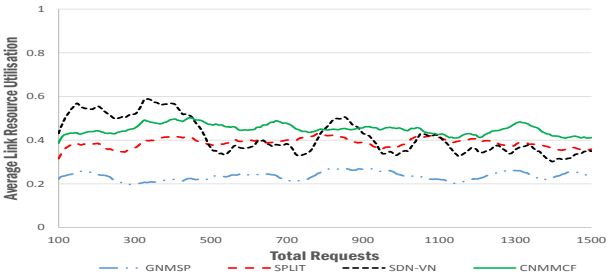

Fig. 5. Average Link Resources Utilization

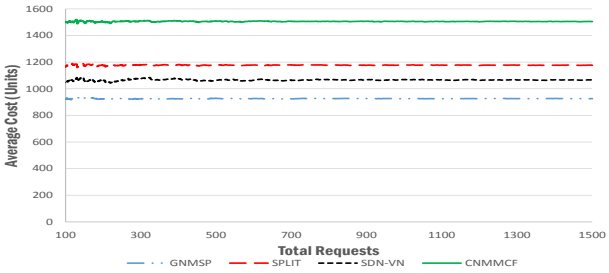

Fig. 6. Average VN Mapping Cost

TABLE I. COMPARED ALGORITHMS

Code	Resource Mapping Method
GNMSP	Greedy Node Mapping and $k-$Shortest Path for links [17]
CNMMCF	Coordinated Node and MCF for Link Mapping [24]
SPLIT	Greedy Node Mapping and path splitting for Link Mapping [7]
SDN-VN	Our Proposal

average switch utilization as the average proportion of the total substrate switch capacity that is under use at any given time. These two parameters are a measure of how efficiently the substrate resources are being utilised.

3) VN Mapping Cost: We define the average VN mapping cost as the total amount of resources that are used by all VNs mapped by a given SN divided by the number of mapped VNs. The total amount of resources used by a VN is obtained by determining and summing the resource utilisation for each virtual link as defined in equation (1). It is a dynamic parameter that takes into account the changes to the mapping cost of the virtual links attached to the VN, and is a measure of how effectively the flow migrations minimize the total resource costs of a VNs.

C. Comparison with other approaches

To evaluate the performance of our proposal, we compare it with three other approaches defined in Table I. Both GNMSP and CNMMCF do not perform any path migrations after the initial mapping, even though GNMSP maps links in one path while CNMMCF allows for paths splitting. SPLIT performs path migrations based only on the loading of links and allows for splitting of paths.

D. Discussion of Results

From graph Fig. 3 it is evident that SDN-VN achieves a significantly higher acceptance ratio than all the other approaches. With respect to the static approaches GNMSP and CNMMCF, this can be attributed to advantages of carrying out path migrations that do only optimize resource usage by existing VNs, but also balances the loading of the SN, giving it a higher chance to accept new requests. With regard to SPLIT, the difference in performance is expected since the path migrations in SPLIT do not consider both switch loadings as well as VN mapping costs. In addition, path splitting uses up more switch memory in SPLIT, which is likely to lead to more failed mappings due to depleted switch resources. It is also worth noting that CNMMCF achieves an almost similar acceptance ratio compared to SPLIT. This is surprising since CNMMCF is a static resource allocation approach. However, it could be explained by the superior node mapping phase (the coordinated node and link mapping) in CNMMCF, which makes it use switch resources better than SPLIT.

In Figs. 4 and 5, we show the average utilisation of both switch and link resources. Except for GNMSP which performs relatively poorly, the other three approaches achieve a comparable utilisation of network resources. The poor utilization of GNMSP can be attributed to its static nature, coupled with the fact that both its node and link mapping algorithms are inferior compared to those of, say, CNMMCF. The fact that SDN-VN and SPLIT have comparable resource utilization profiles is expected since both of them carry out path migrations which allow the substrate network ability to effectively utilize its resources. Once more, while surprising, the fact that CNMMCF performs comparable to SPLIT and SDN-VN could be explained by a superior initial node mapping algorithm. It is however worth remarking that while these three approaches utilise almost the same amount of substrate resources, SDN-VN uses these resources to accept almost 35% more VN requests than SPLIT and CNMMCF which further underlines the effectiveness of our proposal. As part of future work, we will investigate the actual cause of the performance of CNMMCF, say, by utilizing the same node mapping algorithm for all four approaches.

Finally, Fig. 6 shows the average amount of substrate

resources used to map each VN. The fact that CNMMCF has the worst cost is not surprising since it is static, not taking advantage of VN departures to reduce link and switch mapping costs. On the other hand, the best mapping costs for GNMSP are a direct result of the fact that most VN requests are not accepted, resulting from the link mapping algorithm. We also observe that SDN-VN performs slightly better than SPLIT. This is due to the fact that the link migrations in SDN-VN are mindful of the switch resources, in addition to the fact that SDN-VN saves some switch resources since all virtual links are represented by a single flow.

VI. Conclusion

This paper has proposed a dynamic resource management approach which is aware of the both link and switch resources in an SDN based virtualized network environment. We extended a floodlight controller to add a module that, based on a particular situation of the networks, performs path migrations by adding, modifying or deleting flow rules. We have shown through simulations that our proposal performs better than a static approach as well as a dynamic approach which only considers link resources, by accepting 40% more VN requests, and achieving a 10% lower resource cost to the VNs.

However, the results presented is this paper are only an initial step towards an efficient dynamic RM solution in SDN-based virtualized networks. In particular, since mapping of VNs onto a SN is computationally intractable (even when the switches have already been mapped), frequently performing link migrations significantly loads the controller. Our next steps will include evaluating the actual extra loading to the controller, the effect of having multiple controllers, and proposing an initial mapping algorithm that takes into account the memory capacities of all switches along a substrate path, so that remappings are minimized. We also intend to subject the VNs to real traffic using iperf [25] and traffic matrices from geant [20] so as to evaluate throughput, and latency. Finally, it will be interesting to make a real implementation, say based on Flowvisor [4], to evaluate more realistic scenarios.

Acknowledgment

This work is partly funded by FLAMINGO, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme, project TEC2012-38574-C02-02 from Ministerio de Economia y Competitividad, the CONACYT FOMIX project TAMPS-2012-C35-185768 and CONACYT-CDTI Project 189413.

References

[1] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862 – 876, 2010.

[2] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks, 2014.

[3] D. Drutskoy, E. Keller, and J. Rexford. Scalable network virtualization in software-defined networks. *Internet Computing, IEEE*, 17(2):20–27, March 2013.

[4] R. Sherwood et. al. Carving research slices out of your production networks with openflow. *SIGCOMM Comput. Commun. Rev.*, 40(1):129–130, January 2010.

[5] N.C. Fernandes and O.C.M.B. Duarte. Xnetmon: A network monitor for securing virtual networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, June 2011.

[6] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, Fourth 2013.

[7] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM. 25th IEEE International Conference on Computer Communications.*, pages 1–12, 2006.

[8] F. Zane, G. Narlikar, and A Basu. Coolcams: power-efficient tcams for forwarding engines. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 42–52 vol.1, March 2003.

[9] F. Gu, M. Peng, S. Khan, A Rayes, and N. Ghani. Virtual network reconfiguration in optical substrate networks. In *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013*, pages 1–3, March 2013.

[10] R. Mijumbi, J.L. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latre. Design and evaluation of learning algorithms for dynamic resource management in virtual networks. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, NOMS2014, 2014.

[11] K. Kannan and S. Banerjee. Compact TCAM: Flow Entry Compaction in TCAM for Power Aware SDN. In D. Frey, M. Raynal, S. Sarkar, R. K. Shyamasundar, and P. Sinha, editors, *ICDCN*, volume 7730 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 2013.

[12] T. Feng, J. Bi, and K. Wang. Joint allocation and scheduling of network resource for multiple control applications in SDN. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–7, May 2014.

[13] S.A Shah, J. Faiz, M. Farooq, A Shafi, and S.A Mehdi. An architectural evaluation of sdn controllers. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3504–3508, June 2013.

[14] R. Trivisonno, I. Vaishnavi, R. Guerzoni, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani. Virtual links mapping in future sdn-enabled networks. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, pages 1–5, Nov 2013.

[15] D. Mehmet and A. Mostafa. Design and analysis of techniques for mapping virtual networks to software-defined network substrates. *Computer Communications*, 45(0):1 – 10, 2014.

[16] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. VNR Algorithm: A Greedy Approach for Virtual Networks Reconfigurations. In *GLOBECOM*, pages 1–6. IEEE, 2011.

[17] M. Yu, J. Yi, Y.and Rexford, and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, March 2008.

[18] K. Kannan and S. Banerjee. Compact TCAM: Flow Entry Compaction in TCAM for Power Aware SDN. In D. Frey and M. Raynal, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 439–444. 2013.

[19] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and R. P. Ligia. Using mininet for emulation and prototyping software-defined networks. In *Communications and Computing (COLCOM), IEEE Colombian Conference on*, pages 1–6, June 2014.

[20] GéANT: the pan-European research and education network. http://www.geant.net/Pages/default.aspx. Accessed: 2014-07-25.

[21] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS '01, pages 346–353, Washington, DC, USA, 2001. IEEE Computer Society.

[22] C. Barnhart, N. Krishnan, and P. Vance. Multicommodity Flow Problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2354–2362. Springer US, 2009.

[23] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/about/. Accessed: 2014-07-13.

[24] M. Chowdhury, M.R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, 20(1):206 –219, feb. 2012.

[25] H. Chung-hsing and K. Ulrich. IPERF: A Framework for Automatic Construction of Performance Prediction Models. In *In Workshop on Profile and Feedback-directed Compilation (PFDC)*, 1998.