A Scalable SDN Slicing Scheme for Multi-domain Fog/Cloud Services

Roberto Bruschi[2], Franco Davoli[1,2], Paolo Lago[1,2] and Jane Frances Pajo[1,2]
[1]DITEN – University of Genoa, Genoa, Italy
[2]CNIT – Research Unit of the University of Genoa, Genoa, Italy
roberto.bruschi@cnit.it, franco.davoli@unige.it, {paolo | jane.pajo}@tnt-lab.unige.it

***Abstract*—By bringing Cloud-like services much closer to end-users and their devices, Fog computing is foreseen to expand the scope of overlay networks, encompassing different heterogeneity dimensions (i.e., size, geographical distribution, devices/virtual objects (VOs) involved, quality of service (QoS) requirements, etc.). Although highly flexible solutions like Software-Defined Networking (SDN) have been conceived to handle such heterogeneity in future networks, scalability is still an open issue, especially with respect to Fog computing requirements. In this paper, we propose an SDN-based network slicing scheme for supporting multi-domain Fog/Cloud services, which offers high scalability, among other aspects, over legacy ones. Results show that the number of unicast forwarding rules needed to be installed in an overlay drops by up to over one order of magnitude and 4 times compared to the "fully-meshed" and OpenStack cases, respectively, at the cost of possible path sub-optimality, albeit knowledge on the datacenter topology can be used for VO placement optimization.**

I. Introduction

With the onset of Fog computing, Cloud-like services will soon become widely available much closer to end-users and their devices. Fog nodes are expected to be deployed practically anywhere – in street cabinets [1], in micro- and container-based datacenters [2], in mobile base stations [3], etc. – wherever there are available computing/storage resources and of course, network connectivity [4]. By bringing such levels of consumer proximity, the Fog paradigm is foreseen to pave the way towards a hyper-connected world.

The scope of virtual networks is expected to grow with the proliferation of smart devices (e.g., smartphones/tablets, home appliances, wearables, cars, etc.), sensor networks and their interplay with Fog-/Cloud-hosted virtual objects (VOs) [5]; VOs can be understood as the virtual counterpart of physical devices, as well as complementary functionalities such as computing/storage. Virtual networks – that range from enterprise to personal networks – are traditionally realized via tunneling-based overlays on top of a shared telecommunications infrastructure. In a multi-tenant context, they can be viewed as "network slices" [6] that are isolated from each other, providing multiple degrees of freedom to the tenants in defining their respective architectures. However, the increasing heterogeneity among them (i.e., size, geographical distribution, devices/VOs involved, quality of service (QoS) requirements, etc.) entails highly customized operations and complex network management that depose classical paradigms and protocols.

Emerging "softwarization" solutions aim at providing flexibility and programmability levels that could future-proof the network. For instance, the Software-Defined Networking (SDN) paradigm [7] seeks to overcome the infrastructure ossification problem by decoupling network intelligence from the forwarding plane; this renders forwarding devices simple and programmable via an open interface like OpenFlow (OF) [8]. With SDN in place, network management complexity is expected to significantly drop, although scalability is still an open issue.

Moreover, the majority of OF-based approaches in the scientific literature today focuses on efficient datacenter management (e.g., [9]–[11], among others); they usually consider a single datacenter rather than the cooperation of multiple geographically distributed datacenters and, hence, their applicability in supporting multi-domain Fog/Cloud services are presumably limited.

To this end, we propose an SDN-based network slicing scheme that offers the following advantages over legacy mechanisms:

- overlay isolation through tunnel-less communications and non-overlapping OF rules;
- intrinsic support for distributed computing facilities through overlay connectivity inside and among Telco (in-network) datacenters;
- high compatibility/low technological requirements through the use of simple OF functions and layer 2 (L2) addressing criteria, and;
- high scalability through significantly minimizing the number of OF rules in the overlay implementation.

The remainder of this paper is organized as follows. Section II describes the overlay network connectivity and L2 addressing adopted, while frame forwarding rules are discussed in Section III. Numerical results are then presented in Section IV, and finally, conclusions are drawn in Section V.

II. Overlay Network Description

In this work, we consider a scenario with geographically distributed Internet services, where each service can consist of a series of software components (e.g., VOs) that define a "service chain" [12]. Network slicing enables isolation among such services, as well as the logical connectivity among their respective components; if needed, services with complex connectivity can be handled through multiple slices.

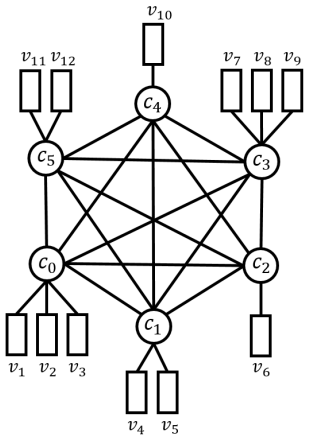


Fig. 1: High-level view of an overlay network.

Each slice $\delta \in \Delta$ is associated to an overlay network Q_δ, specifically designed to provide an L2 interconnection among all the VOs $v \in V_\delta$ in the Telco (in-network) datacenters D. Generally, VOs in V_δ could be hosted by any datacenter $d \in D$.

The overlay Q_δ is organized into N "center nodes," $\{c_0, \ldots, c_{N-1}\}$. Each center node is meant to be mapped on one of the datacenter gateway switches in the infrastructure. For the sake of easy presentation, but without losing generality, in the following we will assume each datacenter with a single gateway switch i_d, $\forall d \in D$, on which different overlay centers can be mapped. A high-level view of an overlay network is illustrated in Fig. 1.

Each VO v is associated to a single center, and all VOs associated to the same center c_n define the set $V_{\delta,n} \subseteq V_\delta$. If $V_{\delta,n}$ is made running in the datacenter d^*, then c_n is mapped on its gateway switch i_{d^*}, giving $i_{d_n} = i_{d^*}$, $d^* \in D : c_n \mapsto i_{d^*}$.

A. Overlay Connectivity

L2 connectivity among Q_δ end-points is built based on shortest-path trees, paths and edges.

In more detail, the shortest-path tree $SPT(r, L)$, whose root and leaves are given by the vertex r and the set of vertexes L, respectively, is defined by the union of the (shortest) paths $P(l, r)$ from each leaf $l \in L$ to r.

$P(l, r)$ is the optimal sequence of edges and hops from l to r. On each of its hops h, two edges e_i, $e_o \in P(l, r)$ are defined, with i/o indicating the flow direction (i.e., towards the leaf/root, respectively). Edges are mapped to distinct ports in each switch i; hence, in such context, ports and edges are used interchangeably hereinafter.

Inside each datacenter $d \in D$, VOs of the same overlay center c_n are interconnected according to the shortest-path tree $SPT(i_d, V_{\delta,n})$, with the root being the gateway switch i_d and the leaf nodes being the VOs $v \in V_{\delta,n}$.

$$SPT(i_d, V_{\delta,n}) \triangleq \cup_{\forall v \in V_{\delta,n}} \{P(v, i_d)\} \tag{1}$$

The intermediate nodes are given by the subset $\hat{I}_{d_n}$ of OF switches available in the datacenter d, $\hat{I}_{d_n} \subseteq I_d$.

TABLE I: Notation used in the matching fields and action list of OF rules.

Parameter	Description
dl_{dst}	destination MAC address
dl_{src}	source MAC address
out	list of ports where a packet matching the rule has to be sent
p_{in}	switch port where a packet to be matched enters

On the other hand, the interconnection among the centers $c_0, \ldots, c_{N-1}$ of the same overlay Q_δ is constructed by calculating the shortest-path trees $Q_\delta^{c_n}$, with the root being i_{d_n} and the leaf nodes being i_{d_m}, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$.

$$Q_\delta^{c_n} \triangleq \cup_{\forall m \in \{0,\ldots,N-1\}, m \neq n} \{P(i_{d_m}, i_{d_n})\} \tag{2}$$

Obviously, when i_{d_m} coincides with i_{d_n} , $P(i_{d_m}, i_{d_n}) = \emptyset$.

It is worth noting that each virtual link of the aforementioned virtual topologies corresponds to a number of OF rules to be installed for each destination, on every crossed physical switch. More details on these rules will be discussed in the following sections.

B. OpenFlow Notation

One of the main design objectives of the proposed scheme resides in the use of few simple primitives defined by the OF protocol (ref. ver. 1.3.1 [8]) that are widely supported by commercial switches in the market.

It is well-known that most of the functionalities and mechanisms of such protocol are optional, and (if present) their implementation on commercial products is sometimes at the software level. In the latter case, inherent and somewhat unpredictable performance decays may occur; hence, the authors decided to use basic matching and action rules that are mandatory in every node supporting the OF protocol v1.3.1.

In this paper, the OF rules will be presented in the form:

$$\mathbf{p} \rightarrow l_x, \textbf{ if } match_1 \;\&\&\; match_2 \;\&\&\; \cdots \;\&\&\; match_y \Rightarrow \{action_1,\ action_2, \ldots,\ action_z\} \tag{3}$$

The first part reports the priority of the rule, with lower x values indicating higher priority, while the matching fields and the corresponding list of actions are contained in the second part.

As regards the matching fields, exact matches will be represented with the operator '$\equiv$' and the ones with a wildcard mask with '$\equiv_{prefix}$'. The latter is meant to be applied with a mask hitting the overlay and/or center identifiers in the L2 addresses.

All the rules are meant to be placed in the Table 0 of OF switches, i.e., the first flow table in the OF pipeline [8]. Other notation used in the OF rules is defined in Table I.

C. Layer 2 Addressing

A large part of IT virtualization platforms/hypervisors provide the possibility to associate customized locally administered Medium Access Control (MAC) addresses to virtual

network interfaces associated to virtual machines, containers, or even those managed by the guest operating system. Hence, we exploit this capability in order to generate MAC addresses structured to contain information useful for identifying flows inside/among overlays, and in a form suitable to be matched by OF hardware and software switches at high speeds. This idea has been already proposed in the scientific literature for approaches based on source routing [10] and tag switching [11] paradigms.

In this work, the rules are designed based on the matching of IEEE Ethernet 48-bit MAC addresses [13] and optionally on IEEE 802.1Q Virtual LAN (VLAN) tags [14], whose possible mapping with the overlay network addressing is illustrated in Fig. 2.

Particularly, the MAC address is organized into three fields, namely (from the most to the least significant) the Overlay, the Center, and the Host Identifiers (IDs). If only the MAC address is used for the overlay network addressing the Host ID is kept to a size of 2 Bytes, the Center ID to 1 Byte, and the Overlay ID to 22 bits (i.e., 3 Bytes minus the 2 flag bits for universal/local (U/L) and individual/group (I/G) addresses). If a VLAN tag is also used, the sizes of the above IDs pass to 3 Bytes, 1 Byte and 34 bits, respectively. Other configurations and sizes may also be supported, but the ones selected are particularly convenient for simple OF matches (OF defined that 48-bit MAC addresses can be matched with masks of lengths from 1 to 6 Bytes at a step of 1 Byte, and only precise matching of VLAN tags are supported).

All VOs belonging to the same overlay will have the same value on the Overlay ID, and all those bound to the same center will share also the same Center ID.

III. Frame Forwarding Rules

Let us consider the set of (in-network) datacenters $\tilde{D} \subseteq D$ that host VOs of the overlay network Q_δ. The set $\tilde{D}$ is identified as all the datacenters on whose gateway node a Q_δ center has been mapped i.e., $\tilde{D} \triangleq \{\forall d \in D : \exists c_n \mapsto i_d\}$.

A. Unicast Forwarding

An example of datacenter internal connectivity is illustrated in Fig. 3. For each VO v hosted in d, the following couple of OF rules are installed on each $h \in P(v, i_d)$, and only the first rule on the root i_d:

$$\mathbf{p} \rightarrow l_1, \ \textbf{if } dl_{dst} \equiv addr(v) \Rightarrow out \rightarrow e_i \qquad \text{(A)}$$

$$\mathbf{p} \rightarrow l_3, \ \textbf{if } dl_{src} \equiv addr(v) \Rightarrow out \rightarrow e_o \qquad \text{(B)}$$

As can be noted, rules (A) and (B) perform a precise matching of the L2 address of v on the destination and the source L2 addresses, respectively. Particularly, the rule (B) allows to move frames generated by v (that consequently have L2 source address of v) towards the i_d node, independently of their destination.

If a frame generated by v is directed to a another VO $\hat{v} \in V_{\delta,n}$, in the first interconnection switch i^* where $P(v, i_d)$ and $P(\hat{v}, i_d)$ intersect, the frame will hit both rules (A) and (B), i.e.:

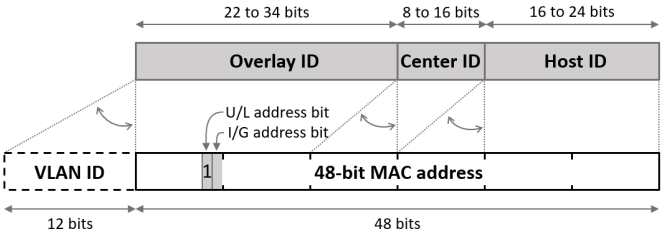


Fig. 2: Overlay network addressing scheme and possible mapping on IEEE Ethernet 48-bit MAC addresses and IEEE 802.1Q VLAN tags.

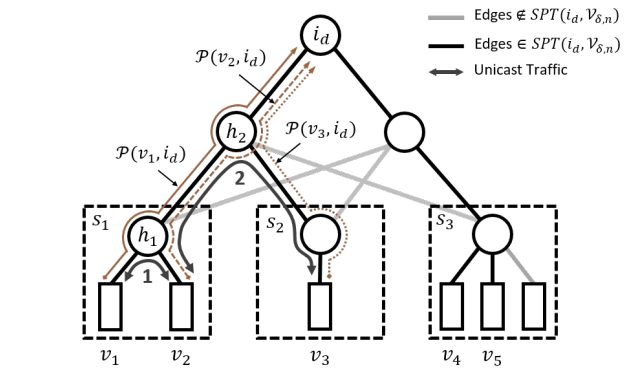


Table 0 of h_1

Priority	Match	Action
l_1	$dl_{dst} \equiv addr(v_1)$	$out \rightarrow e_{i,1}$
l_1	$dl_{dst} \equiv addr(v_2)$	$out \rightarrow e_{i,2}$
⋮	⋮	⋮
l_3	$dl_{src} \equiv addr(v_1)$	$out \rightarrow e_{o,1}$
l_3	$dl_{src} \equiv addr(v_2)$	$out \rightarrow e_{o,2}$

Table 0 of h_2

Priority	Match	Action
l_1	$dl_{dst} \equiv addr(v_1)$	$out \rightarrow e_{i,1}$
l_1	$dl_{dst} \equiv addr(v_2)$	$out \rightarrow e_{i,2}$
l_1	$dl_{dst} \equiv addr(v_3)$	$out \rightarrow e_{i,3}$
⋮	⋮	⋮
l_3	$dl_{src} \equiv addr(v_1)$	$out \rightarrow e_{o,1}$
l_3	$dl_{src} \equiv addr(v_2)$	$out \rightarrow e_{o,2}$
l_3	$dl_{src} \equiv addr(v_3)$	$out \rightarrow e_{o,3}$

Fig. 3: Example of internal overlay connectivity and unicast packet forwarding among its VOs in the same datacenter.

$$\mathbf{p} \rightarrow l_1, \ \textbf{if } dl_{dst} \equiv addr(\hat{v}) \Rightarrow out \rightarrow e_i$$

and

$$\mathbf{p} \rightarrow l_3, \ \textbf{if } dl_{src} \equiv addr(v) \Rightarrow out \rightarrow e_o$$

For instance, as shown in Fig. 3, this happens in the switches h_1 and h_2 for the frames generated by v_2 that are destined to v_1 and v_3, respectively. However, thanks to its higher priority, rule (A) will be selected, and the frame directly redirected to $\hat{v}$, i.e., v_1 and v_3 in the example.

If the destination is not in the datacenter (and, consequently, the destination L2 address is unknown to all the interconnection switches), frames will eventually reach i_d, and from that node on, their forwarding will be driven by the algorithm of the overlay backbone.

As previously sketched, the backbone connectivity is realized through a fully-meshed overlay among the centers $c_0, \ldots, c_{N-1}$. Differently from the forwarding criterion inside the datacenter, and in order to increase the scalability of the forwarding rules in the backbone, the proposed algorithm does

not rely on the "precise" matching of the L2 addresses, but only on the matching of their prefix containing the Overlay and Center IDs. This choice obviously allows reducing the number of required forwarding rules on the OF switches in the backbone, especially in the case where each center groups a significant number of VOs.

Thus, $\forall c_n \in \{c_0, \ldots, c_{N-1}\}$, the algorithm works by obtaining the spanning tree $Q_\delta^{c_n}$, according to Eq. (2). For each node $i \in Q_\delta^{c_n}$, $i \not\equiv i_{d_n}$, the following rule is configured:

$$\mathbf{p} \rightarrow l_3, \ \textbf{if} \ dl_{dst} \equiv_{prefix} id(\delta, c_n) \Rightarrow out \rightarrow e_o \qquad \text{(C)}$$

Rule (C) is aimed at allowing unicast traffic from any other centers $i_{d_m} \in Q_\delta^{c_n}$ to reach i_{d_n}. Here, the $Q_\delta^{c_n}$ tree is crossed in upstream from the leaves i_{d_m} (possible source of traffic) to the root node i_{d_n} (the gateway switch of the datacenter containing the VO to which the traffic is addressed).

It is important to note that for wide-area unicast forwarding, this concept of using the tree $Q_\delta^{c_n}$ rooted at the destination gateway switch yields simpler rules, since there is only one path (and port) on the source that is part of $Q_\delta^{c_n}$. Then, from i_{d_n} on, the switching will be driven by the rules given by the algorithm for datacenter internal connectivity. Furthermore, since the optimal path is always chosen when crossing a tree in upstream, the sub-path from a VO in one datacenter to the gateway switch of another datacenter is always optimal.

Possible path sub-optimality in the proposed approach may arise when trees are crossed in downstream for the case of asymmetric edge weights, as well as in datacenters with high levels of path diversity.

B. Broadcast/Multicast Forwarding

Although broadcast/multicast forwarding is left for future work, it is important to remark that the MAC broadcast/multicast address will be re-mapped to the overlay broadcast/multicast address, and vice versa, at the hypervisor switch, and the I/G bit is set to '1'. Moreover, broadcast and multicast forwarding rules will be almost identical, with a slightly different matching field values; the priority l_2 is reserved for such rules.

IV. Numerical Results

The performance of the proposed approach is numerically evaluated through a series of Matlab simulations. Ten runs of varying seeds are executed for tests with random nature in order to establish confidence in the results.

First, we take a look at the number of unicast forwarding rules needed to be installed in an overlay, comparing our overlay network (ON) approach with two baselines, the fully-meshed (FM) and OpenStack (OS) [15] cases.

In FM, we suppose to install rules with exact matches for each source/destination pair of VOs in an overlay, on all switches along the optimal paths between them. We consider its best-case scenario where all the VOs in the same datacenter are hosted in the same server, minimizing the number of rules for simplicity and for the sake of comparison with the other cases. Exact matching rules for each source/destination pair

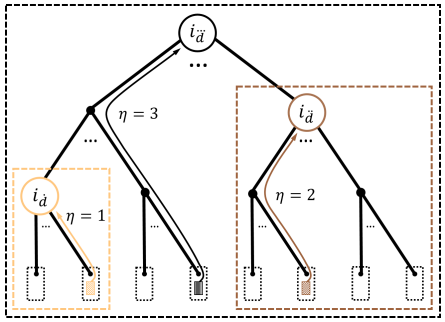


Fig. 4: Different datacenter depths $\dot{d}$, $\ddot{d}$ and $\dddot{d}$.

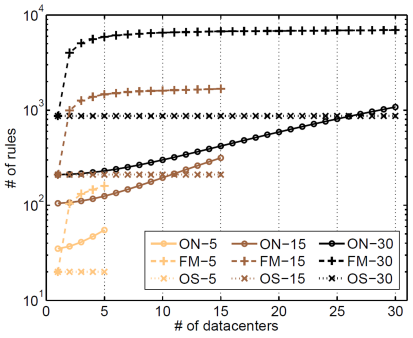
are also considered in OS, but they are only installed on the hypervisor switches involved in an overlay. This concept tries to take into account the scenario where legacy routers and/or switches are in place instead of SDN-enabled switches, using conventional routing/tunneling protocols on top of the OpenStack platform.

Then, we evaluate the path lengths between two VOs, calculated using Dijkstra-based shortest path (SP) and the proposed ON algorithms. The comparison is built on a conservative assumption that conventional frame forwarding mechanisms use the optimal/shortest path. Moreover, we focus on the case when both VOs are inside the same datacenter to get a view on the possible path sub-optimality of our approach. Equal (eqW) and random asymmetrical (rndW) edge weights are considered for different datacenter topologies.

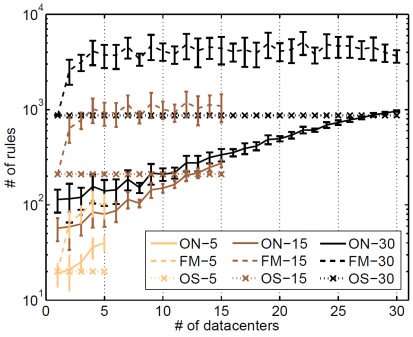
A. Number of Unicast Forwarding Rules

In this study, we define three datacenter depths $\dot{d}$, $\ddot{d}$ and $\dddot{d}$ sorted according to the number of hops η from the VO to the datacenter gateway switch, as shown in Fig. 4. A datacenter of depth $\dddot{d}$ has $\eta = 3$ that corresponds to the traditional three-layer datacenter topology [16], while those of depths $\dot{d}$ ($\eta = 1$) or $\ddot{d}$ ($\eta = 2$) are supposed to cover the heterogeneity of facilities in a Fog scenario. For the sake of simplicity, but without loss of generality, the topology of the backbone is not considered in this evaluation, assuming a full-meshed connectivity between datacenters.

Initially, we impose that all datacenters involved in an overlay are with depth $\dddot{d}$, corresponding to a traditional Cloud scenario. The number of unicast forwarding rules are then calculated for 5, 15 and 30 VOs, uniformly distributed among 1 up to 30 datacenters. Fig. 5a shows that for 5 VOs OS has less rules, although it is important to recall that there are additional routing/tunneling costs incurred. On the other hand, the best-case scenario of FM is only better than our approach in the case of 1 datacenter involved, since it coincides with OS having rules only in the hypervisor switch. As the number of VOs in the overlay increases, it can be observed that our approach has consistently less rules than FM (i.e., by up to over one order of magnitude) and the majority of the OS cases (i.e., by up to over 4 times).



(a) traditional datacenter depth $\dddot{d}$



(b) random datacenter depth

Fig. 5: Number of unicast forwarding rules in the overlay network (ON), fully-meshed (FM) and OpenStack (OS) cases.

Moving to the Fog scenario, we consider the probability mass function $\{P(\eta), \eta = 1, 2, 3\} := \{0.6, 0.3, 0.1\}$. This means that 60%, 30% and 10% of the datacenters involved in an overlay are generated in the simulation to have depths $\dot{d}$, $\ddot{d}$ and $\dddot{d}$, respectively. Fig. 5b shows more or less the same trend as in Fig. 5a. However, it is interesting to note that for 5 VOs, we now have cases where our approach becomes better than OS by introducing smaller datacenters. Error bars indicate the 95% confidence interval for the 10 runs.

B. Path Lengths Inside the Datacenter

Here we consider four datacenter topologies (i.e., Traditional three-layer tree [16], Fat tree [16], Spine-and-Leaf [17] and BCube [18]) interconnecting 16 servers, as illustrated in Fig. 6. The first two are conventional tree-based architectures commonly used in datacenters, while the other two are more recently conceived to better support East/West (E/W) traffic.

For a given topology, we initially place two VOs v_1 and v_2 in server s_1. Then, path lengths are obtained using the SP and ON algorithms, varying the location of v_2 from s_1 through s_{16}. The eqW cases have all edge weights set to '3', while rndW ones have weights drawn from the discrete uniform distribution $\mathcal{U}\{1, 5\}$.

In all four topologies, the paths obtained for both algorithms coincide when all edges in the datacenter have equal weights, as indicated by the **eqW** curves in Fig. 7. It can also be observed that indeed the Spine-and-Leaf and BCube architectures yield shorter path lengths on average than the tree-based ones, demonstrating their suitability for E/W traffic support.

Furthermore, Fig. 7 shows that when the edges have random asymmetrical weights, our approach (i.e., **rndWON** curves) has, on average, higher path lengths compared to SP (i.e., **rndWSP** curves), demonstrating cases of path sub-optimality; differences between the ON and SP path lengths in all the runs are illustrated by the box plots (i.e., **rndWON** − **rndWSP**) for statistical significance. However, it can be observed that, for a given topology, path lengths from s_1 to a subset of servers $\{s_x\}$ correspond or are close to the optimal ones. This implies that knowledge on the datacenter topology can be used to better place VOs of the same overlay in a datacenter.

Although our approach does not fully exploit the path diversity offered by such datacenter topologies, it is important to note that, in a multi-tenant context, centers of different overlays can be mapped to different gateway switches for datacenter load-balancing.

V. Conclusion

The Fog paradigm is foreseen to expand the scope of overlay networks, encompassing different heterogeneity dimensions. Although highly flexible solutions like SDN have been conceived to handle such heterogeneity in future networks, scalability is still an open issue. In this respect, an SDN-based network slicing scheme for supporting multi-domain Fog/Cloud services, which offers high scalability, among other aspects, is proposed.

Overlay isolation is achieved through non-overlapping OF rules that only consider simple (and mandatory) OF functions and L2 addressing criteria. Results show that the number of unicast forwarding rules needed to be installed in an overlay drops by up to over one order of magnitude and 4 times compared to the fully-meshed and OpenStack cases, respectively. On the downside, possible path sub-optimality may occur, albeit knowledge on the datacenter topology can be used for VO placement optimization. Additionally, while the proposed approach does not fully exploit the path diversity offered by datacenter topologies, in a multi-tenant context, mapping centers of different overlays to different gateway switches can provide load-balancing benefits.

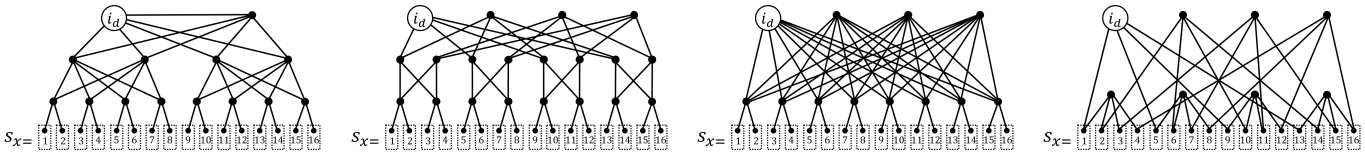
Moving forward, we would like to integrate not only broadcast/multicast forwarding rules, but also a scheme for supporting seamless intra-/inter-datacenter VO migration.

Acknowledgment

This work was supported by the INPUT (In-Network Programmability for next-generation personal cloUd service supporT) project, funded by the European Commission under the Horizon 2020 Programme (Grant no. 644672).

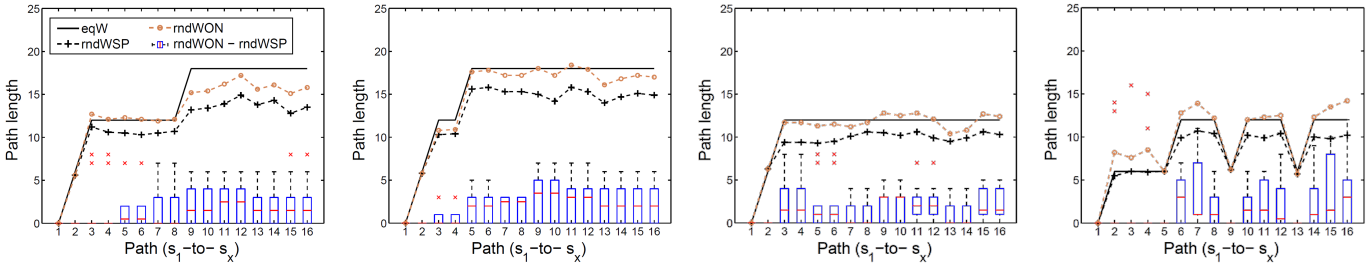
References

[1] M. Yannuzzi, F. van Lingen, A. Jain, O. L. Parellada, M. M. Flores, D. Carrera, J. L. Pérez, D. Montero, P. Chacin, A. Corsaro and A. Olive, "A New Era for Cities with Fog Computing," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 54–67, Mar. 2017.



(a) Traditional three-layer tree (b) Fat tree (c) Spine-and-Leaf (d) BCube

Fig. 6: Different datacenter topologies interconnecting 16 servers.



(a) Traditional three-layer tree (b) Fat tree (c) Spine-and-Leaf (d) BCube

Fig. 7: Path lengths obtained using the shortest path (SP) and overlay network (ON) algorithms for the different datacenter topologies with equal (eqW) and random (rndW) edge weights.

[2] B. Kim and B. Lee, "Integrated Management System for Distributed Micro-Datacenters," in *Proc. of the 18th Int. Conf. on Advanced Communication Technology (ICACT)*, PyeongChang, Korea, Jan. 2016, pp. 466–469.

[3] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach and F. Giust, "Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[4] "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are," White Paper, 2015. [Online]. Available: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[5] M. Nitti, V. Pilloni, G. Colistra and L. Atzori, "The Virtual Object as a Major Element of the Internet of Things: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1228–1240, Q2 2016.

[6] R. Pries, H. J. Morper, N. Galambosi and M. Jarschel, "Network as a Service - A Demo on 5G Network Slicing," in *Proc. of the 28th Int. Teletraffic Congr. (ITC 28)*, vol. 1, Würzburg, Germany, Sep. 2016, pp. 209–211.

[7] B. A. A. Nunes, M. Mendonca, X. Nguyen, K.-N. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, Q3 2014.

[8] "OpenFlow Switch Specifications Version 1.3.1 (Wire Protocol (0x04)," Sep. 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-specv1.3.1.pdf

[9] R. Kawashima and H. Matsuo, "Non-Tunneling Edge-Overlay Model using OpenFlow for Cloud Datacenter Networks," in *Proc. of the 5th IEEE Int. Conf. on Cloud Computing Technology and Science (CloudCom)*, Bristol, UK, Dec. 2013, pp. 176–181.

[10] A. Hari, T. V. Lakshman and G. Wilfong, "Path Switching: Reduced-State Flow Handling in SDN Using Path Information," in *Proc. of the 11th ACM Int. Conf. on Emerging Networking Experiments and Technologies (CoNEXT15)*, Heidelberg, Germany, Dec. 2015.

[11] A. Schwabe and K. Holger, "Using MAC Addresses as Efficient Routing Labels in Data Centers," in *Proc. of the 3rd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'14)*, Chicago, IL, USA, Aug. 2014, pp. 115–120.

[12] A. Leivadeas, M. Falkner, I. Lambadaris and G. Kesidis, "Resource Management and Orchestration for a Dynamic Service Chain Steering Model," in *Proc. of the 2016 IEEE Global Communications Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[13] "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture," *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*, pp. 1–74, Jun. 2014.

[14] "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks," *IEEE Std 802.1Q-2003*, May 2003.

[15] "OpenStack." [Online]. Available: https://www.openstack.org/

[16] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang and M. F. Zhani, "Data Center Network Virtualization: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, Q2 2013.

[17] "Cisco Data Center Spine-and-Leaf Architecture: Design Overview," White Paper, 2016. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.pdf

[18] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang and S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," in *Proc. of the 2009 ACM SIGCOMM Conf. (SIGCOMM'09)*, Barcelona, Spain, Aug. 2009.