A Multi-Clustering Approach to Scale Distributed Tenant Networks for Mobile Edge Computing

Roberto Bruschi, *Senior Member, IEEE,* Franco Davoli, *Senior Member, IEEE,*
Paolo Lago, and Jane Frances Pajo, *Student Member, IEEE*

***Abstract*—Fifth generation (5G) mobile networks will lead to a deep integration between networks and applications. Through novel paradigms like Network Functions Virtualization (NFV) and Edge Computing, new classes of heterogeneous application services will be enabled to run close to mobile end-user devices with zero-perceived latency and fully-cognitive dynamic reconfiguration capabilities. Such "vertical" applications exhibit diverse performance/scalability requirements, and will rely on highly distributed, extremely virtualized, multi-tenant and software-defined infrastructures. In such context, handling the required operations in a scalable and dynamic fashion will be of paramount importance. A specific aspect, addressed by Software Defined Networking (SDN), regards the provision of suitable communication channels, once resource allocation mechanisms have performed the most efficient deployment of Virtual Network Function (VNF) instances, and VNF chaining needs to be implemented to enable network services. In this respect, this paper introduces the Multi-Cluster Overlay (MCO) network paradigm: a tunnel-less SDN scheme for scalable realization of Virtual Tenant Networks (VTNs) across the 5G distributed infrastructure, able to support (bulk) migrations of software instances among geo-distributed computing resources in a seamless and effective fashion. Numerical simulation and experimental results show that the MCO achieves up to over one order of magnitude smaller number of forwarding rules than other state-of-the-art SDN mechanisms, while also assuring high performance during reconfiguration operations.**

***Index Terms*—Edge Computing, 5G, SDN, Mobility**

I. Introduction

Edge Computing paradigms are gaining momentum in the core 5G technology specification, as they bring cloud-native applications closer to end-users and network-connected "things", enabling new classes of services with challenging performance/operating requirements. This reduction in the networking scope of user applications (or, at least, of some of their application service components) is instrumental to meet the stringent needs of radically new vertical sectors, in the fields of Industry 4.0, e-Health, Smart Cities, and Electrical Grids, among others, to be supported by the upcoming 5G technologies.

Telecommunication infrastructures are evolving into a sort of geo-distributed datacenter with advanced virtualization and computing capabilities, able to host both network and application software instances from multiple tenants. Computing resources will be provided by resource-constrained facilities, placed at various levels of the network [1]–[3].

At the same time, cloud-native applications have been evolving towards software architectures to boost and embrace modularity. State-of-the-art applications are composed by a large number of *Virtual Objects (VOs)*, i.e., "microservices" [4] embedded into various execution containers (e.g., Virtual Machines (VMs), Linux Containers, etc.), and interconnected through virtual network topologies to enable even more complex *Service Chains (SCs)* [5]. An application orchestrator is used to dynamically manage the lifecycle of each software instance and of its virtual connectivity to conform with the application-level workload and requirements [6].

Thanks to their modularity, cloud-native applications are the perfect counterpart to be "vertically deployed" onto 5G-ready infrastructures: VOs can be placed into the geo-distributed computing facilities close to User Equipment (UE) [7], and migrated from a datacenter to another as UEs move, in order to provide seamless user experiences. Moreover, depending on the nature of the application, VOs could be differently tolerant to the end-to-end latency, and hence, to their proximity to UEs. Thus, it is reasonable to assume that each VO is associated to a Service Level Agreement (SLA), defining its Quality of Service (QoS) requirements, which can be used to select its placement in the network. Obviously, the tighter the SLA proximity requirement of a VO is, the more frequently it might be migrated among datacenters as the UEs move, causing infrastructure reconfigurations. In addition, users can access multiple Edge applications, which might be provided by different tenants.

The role of Software-Defined Networking (SDN) technologies [8], such as *OpenFlow (OF)* [9], becomes crucial in this challenging scenario. SDN not only offers effective and flexible means for isolating Virtual Tenant Networks (VTNs), but also the possibility of supporting SC reconfigurations in an effective and seamless fashion. Particularly, during VO migrations, SDN is well-known to fit re-routing mechanisms that would avoid any network-induced performance drawbacks (e.g., packet losses [10], delays due to reactive rebuilding of switching tables [11], encapsulation overhead [12]), which typically occur in legacy Layer-2/3 networks [13].

In this paper, we propose the *Multi-Cluster Overlay (MCO)*, an SDN-based mechanism specifically designed to realize wide-area VTNs and effectively support dense deployments of mobile VOs at the network edge in a highly scalable fashion,

Manuscript received April 20, 2018; revised January 7, 2019.

R. Bruschi and P. Lago are with the National Laboratory of Smart, Sustainable and Secure Internet Technologies and Infrastructures (S3ITI) of the Italian National Consortium for Telecommunications (CNIT), Genoa, Italy (e-mail: roberto.bruschi@cnit.it, paolo@tnt-lab.unige.it).

F. Davoli and J. F. Pajo are with the Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN) of the University of Genoa, and with the CNIT S3ITI National Laboratory, Genoa, Italy (e-mail: franco.davoli@unige.it, jane.pajo@tnt-lab.unige.it).

TABLE I: Main acronyms.

BN	Back-end Network	PN	Personal Network
L2	Layer-2	SC	Service Chain
MCO	Multi-Cluster Overlay	VO	Virtual Object
OF	OpenFlow	VTN	Virtual Tenant Network

as well as to relax any (resource/functional) requirements at the SDN switches in the network infrastructure. As better discussed in the remainder of this paper, the MCO provides the following main advantages with respect to state-of-the-art SDN mechanisms: **(i)** overlay isolation through tunnel-less communications [14] (i.e., non-overlapping OF rules among different overlays without the use of resource-hungry tunneling protocols); **(ii)** intrinsic support for distributed computing facilities (i.e., overlay connectivity is provided inside and among datacenters at the network edge); **(iii)** clustering of VOs with similar SLA requirements to boost scalability through VO aggregation, and to enable a more abstracted and agile overlay network control; **(iv)** efficient network support for single and bulk VO seamless live migrations (i.e., no network-induced packet loss); **(v)** low infrastructure-level requirements (i.e., simple and mandatory OF filters and actions, as well as Layer-2 (L2) addressing criteria); **(vi)** high performance (i.e., optimal/close-to-optimal traffic paths, low computational overhead); and **(vii)** high scalability, by significantly reducing the number of OF rules (and entries in the switch forwarding tables) in the overlay implementation, as well as the number of rule updates in case of bulk VO migrations.

An initial version of this work was presented in [15], in which the concept of VO clusters is first introduced, and base unicast connectivity is discussed. Here, we further advance the contribution by incorporating rules for broadcast/multicast forwarding, fixed and mobile access terminations, as well as for seamless migrations of VOs. A use case on virtualized *Personal Networks (PNs)* under the framework of the INPUT project [16] is also presented, and considered as basis for an experimental testbed and experimental results.

The remainder of this paper is organized as follows. Section II introduces related work and highlights the paper positioning in this context. Section III describes the MCO L2 connectivity and addressing schemes. Section IV and Section V discuss the MCO OF forwarding rules, and the rule updates for seamless migrations, respectively. Section VI introduces the metrics considered in the evaluations. Numerical and experimental results are then presented in Sections VII and VIII, respectively. Finally, conclusions are drawn in Section IX. For a quick guide on the main acronyms used throughout the paper, refer to Table I.

II. Related Work

As regards the efficient management of VOs and traffic steering in SDN, a large part of related work (e.g., [14], [17]–[20], among others) focused on a single datacenter. In this respect, the MCO approach provides a broader scope, by explicitly addressing multiple geo-distributed datacenters and their wide-area interconnection (an essential characteristic in the Mobile Edge Computing environment).

In the framework of Network Functions Virtualization (NFV) and Service Functions Chaining (SFC), there is a vast literature considering the problems of *Resource Allocation* and *Network Functions Placement*. In particular, [21]–[27] deal with resource allocation and VM placement, also with respect to QoS and SLA compliance. More recent works [28]–[33] treat the problem of Virtual Network Function (VNF) placement and chaining, taking into account various characteristics for optimality, and also considering user mobility in some cases. In particular, [32] studies the VNF placement problem for the optimal SFC formation across geographically distributed clouds, and models both link delays and computational delays in the formulation of the optimization problem. Further works [34]–[39] extend the consideration to the specific mobile wireless networking environment, also in its evolution toward 5G and network slicing.

It is worth noting that the MCO approach we consider here is complementary to the placement and chaining problems. By exploiting SDN configurability, we address the provision of suitable communication channels, once resource allocation mechanisms have performed the most efficient deployment of VNF/VO instances, and VNF/VO chaining needs to be effected to implement network services. In other words, we assume the deployment of VNFs/VOs to have been set in place by means of the above-cited optimization mechanisms, and we aim to provide their interconnection within and among the distributed datacenters, in order to allow any required chaining and the capability of maintaining the connectivity whenever edge resources need to be migrated to follow the user mobility and comply with the resource proximity imposed by QoS requirements. Though in a different context, the problem we address presents some similarity with the server selection considered in [40], where the selection problem is transformed into an optimal routing one.

III. Multi-Cluster Overlay Networks

We consider a scenario where users have access to physical/virtual objects through geo-distributed VTNs and the set of datacenters D in the network. Objects are accessed through their physical (fixed/mobile) and virtual network endpoints.

A tenant $\delta \in \Delta$ is associated to a VTN, implemented as an MCO network Q_δ – a sort of private overlay specifically designed to provide L2 interconnection among its endpoints, including the user premises physical equipment deployed in the home ($\hbar_u$ – which can be understood as the gateway to a fixed home/enterprise network), and/or connected through mobile access (set $\mathcal{M}_\delta$), as well as the set of VOs (V_δ) hosted within a subset of datacenters ($D_\delta \subseteq D$).

Endpoints in Q_δ are given by $F(Q_\delta) \triangleq \{\varphi(\upsilon), \forall \upsilon \in \hbar_u \cup \mathcal{M}_\delta \cup V_\delta\}$, where $\varphi(\upsilon)$ is a function providing the current position of the (physical/virtual) object υ as a switch-port pair. It is worth noting that the endpoints are terminations both towards the network edge and towards execution containers in D. Generally, a VO v can be hosted by any datacenter $d \in D$, and VOs must be able to migrate across servers of the same datacenter, or across different datacenters.

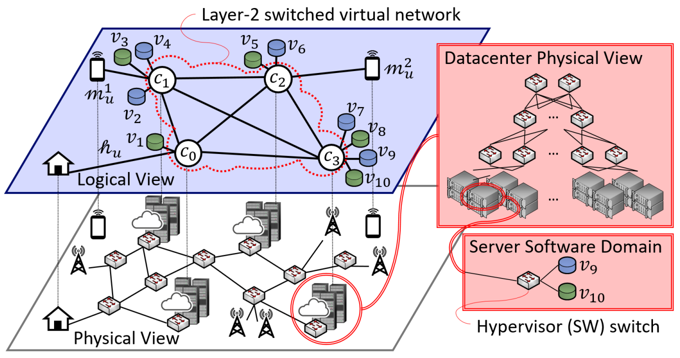

Layer-2 switched virtual network
Datacenter Physical View
Server Software Domain
Hypervisor (SW) switch
Logical View
Physical View
v_1 … v_{10}, c_0, c_1, c_2, c_3, m_u^1, m_u^2, $\hbar_u$

Fig. 1: Physical and logical view of an MCO network owned by a single user.

To reflect the peculiarities above, Q_δ is organized into N clusters of VOs that are identified with *cluster centers*, $\{c_0, \dots, c_{N-1}\}$ – each one mapped to a datacenter gateway switch in the Telecom infrastructure. For instance, Fig. 1 illustrates the physical and logical view of an MCO network owned by user u, having 4 cluster centers and 13 endpoints (i.e., 10 VOs, home $\hbar_u$ and mobile access $\mathcal{M}_\delta = \{m_u^1, m_u^2\}$ terminations).

The set $V_{\delta,n} \subseteq V_\delta$ corresponds to the VO cluster bound to the center c_n. If $V_{\delta,n}$ is hosted in the datacenter d, then c_n maps to its gateway switch i_d (i.e., $i_{d_n} \equiv i_d$, $d \in D_\delta : c_n \mapsto i_d$). Without loss of generality, we assume henceforth that each datacenter has a single gateway switch i_d, $\forall d \in D$, on which (multiple) centers can be mapped, in order to simplify the representation.

Although each VO v belongs to a single cluster in Q_δ, it is important to note that other VO interfaces may be associated with cluster centers of *Back-end Networks (BNs)*, which are essential for some data handling operations. Without loss of generality, we will focus on Q_δ's connectivity in the discussion for easy presentation, but PN-BN interactions will be covered in the experiments.

A. Overlay Connectivity

We build the L2 connectivity among endpoints in Q_δ on the basis of paths and shortest-path trees, where each physical/virtual OF switch along a path constitutes a hop.

By definition, the shortest-path tree $SPT(r, L)$ is the union of the (shortest) paths $\mathcal{P}(l, r)$ from each leaf $l \in L$ to the root r, where $\mathcal{P}(l, r)$ is the optimal sequence of edges and hops from l to r. For a given $\mathcal{P}(l, r)$, two edges e_i, e_o are defined on each of its hops h, and collectively as $\xi \triangleq \{e_i, e_o\}$. The direction of the flow (i.e., towards the leaf/root) is indicated by i/o, respectively.

Considering that edges are mapped to distinct ports on a switch, the terms *ports* and *edges* will be used interchangeably hereinafter. From the perspective of paths, however, an edge e can also be defined by the pair of vertexes at its endpoints, which is given by the function $vertex(e)$.

Suppose that the datacenter $d \in D_\delta$ hosting the VO cluster bound to the center c_n houses the sets $\mathcal{S}_d$ of servers and I_d of OF switches. Inside d, VOs in the set $V_{\delta,n}$ are interconnected according to the shortest-path tree $SPT(i_{d_n}, V_{\delta,n})$, with the gateway switch i_{d_n} being the root and the VOs $v \in V_{\delta,n}$ being the leaves. The servers and intermediate switches involved are given by the subsets $\mathcal{S}_{d_n}^\delta \subseteq \mathcal{S}_d$ and $I_{d_n}^\delta \subseteq I_d$, respectively.

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

Note that, over time, multiple cluster centers of the same overlay Q_δ may be mapped on the same gateway switch i_d. Consequently, for each datacenter d, the internal connectivity for Q_δ will then involve all VOs $v \in V_\delta^d$, where $V_\delta^d \triangleq \cup_{\forall c_n \mapsto i_d} V_{\delta,n}$, realized by the shortest-path tree $SPT(i_d, V_\delta^d)$. The servers and intermediate switches involved are then given by $\mathcal{S}_d^\delta \triangleq \cup_{\forall c_n \mapsto i_d} \mathcal{S}_{d_n}^\delta$ and $I_d^\delta \triangleq \cup_{\forall c_n \mapsto i_d} I_{d_n}^\delta$, respectively.

Conversely, the interconnection among the centers $c_0, \dots, c_{N-1}$ of Q_δ is given by the shortest-path trees $Q_\delta^{c_n}$, $\forall n \in \{0, \dots, N-1\}$, with the root being i_{d_n} and the leaves being i_{d_m}, $\forall m \in \{0, \dots, N-1\}$, $m \neq n$.

$$Q_\delta^{c_n} \triangleq SPT(i_{d_n}, \{i_{d_m}\}) \tag{2}$$

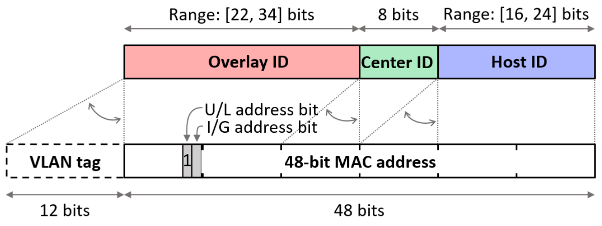


Fig. 2: Overlay network addressing scheme and possible mapping on Ethernet 48-bit MAC addresses and IEEE 802.1Q VLAN tags.

B. Layer-2 Addressing

Today's virtualization hypervisors allows the association of (customized) locally administered MAC addresses to virtual network interfaces. We exploit this capability to configure MAC addresses in a form convenient for flow identification inside/among overlays, as well as for high-speed rule matching in OF hardware (HW) and software (SW) switches.

As illustrated in Fig. 2, the MCO forwarding rules are designed based on the matching of Ethernet 48-bit MAC addresses [41] (and optionally on IEEE 802.1Q VLAN tags [42]). The MAC address is partitioned into three fields – from the most significant bit, there is the 22-bit Overlay ID (i.e., 3 Bytes minus the 2 flag bits for universal/local (U/L) and individual/group (I/G) addresses), the 8-bit Center ID, and the 16-bit Host ID. When a VLAN tag is also used, the Overlay and Host IDs can have lengths within the ranges [22,34] and [16,24] bits, respectively. Such configurations are particularly convenient for simple OF matches, considering that OF 1.3.1 [9] defined matching of 48-bit MAC addresses with 1 to 6 Bytes masks at a step of 1 Byte, and only supports precise matching of VLAN tags.

IV. Frame Forwarding Rules

Consider the subset of datacenters $D_\delta \subseteq D$ that host one or more cluster centers (i.e., $D_\delta \triangleq \{\forall d \in D : \exists c_n \mapsto i_d\}$). The virtual topologies inside and among $d \in D_\delta$ that define the overlay connectivity are given by a number of OF rules installed on every crossed switch, for each VO in V_δ. The OF rules are defined according to two complementary algorithms

TABLE II: Key notations.

Parameter	Description
$\mathcal{P}(l, r)$	the optimal sequence of edges and hops from the leaf l to the root r
e_i (e_o)	the edge at a generic hop $h \in \mathcal{P}(l, r)$ towards l (r)
ξ	the set $\{e_i, e_o\}$ of edges at a generic hop $h \in \mathcal{P}(l, r)$
$vertex(e)$	the function that returns the pair of vertexes at the endpoints of an edge e
$SPT(r, L)$	the shortest-path tree with r being the root and L being the set of leaves, $SPT(r, L) \triangleq \cup_{\forall l \in L}\{\mathcal{P}(l, r)\}$
$\mathcal{S}_d^\delta \subseteq \mathcal{S}_d$	the subset of servers in d that are involved in the MCO network Q_δ
$I_d^\delta \subseteq I_d$	the subset of OF switches in d that are involved in Q_δ
i_{d_n}	the gateway switch i_d of $d \in D_\delta$ on which the cluster center c_n is mapped
$\varphi(v)$	the function that returns the position of the (physical/virtual*) object v as a switch-port pair (i_v, p_v) *a VO is denoted by v
$V_\delta^d \subseteq V_\delta$	the subset of VOs in Q_δ that are hosted in $d \in D_\delta$
$V_{\delta,n} \subseteq V_\delta$	the subset of VOs in Q_δ that are bound to c_n
$Q_\delta^{c_n}$	the wide-area shortest-path tree with the gateway switch i_{d_n} being the root and the other gateway switches i_{d_m}, $m \in \{0, \dots, N-1\}$, $m \neq n$, being the leaves, $Q_\delta^{c_n} \triangleq SPT(i_{d_n}, \{i_{d_m}\})$
p_{in}	the switch port where a frame to be matched enters
out	the list of ports where a frame matching the OF rule has to be sent
dl_{src} (dl_{dst})	the source (destination) MAC address [1]

– one acting inside each datacenter, and the other on the wide-area infrastructure. The rationale behind this division is to isolate as much as possible the number of network reconfigurations during VO migrations or changes in the mapping between overlay and underlay resources. Independently of such implementation, the rules will be described according to their forwarding type (i.e., unicast, broadcast/multicast).

A. The OpenFlow Notation

The MCO is designed to use simple and mandatory primitives defined in the OF 1.3.1 protocol [9] and widely supported by commercial switches.

In this paper, the OF rules are expressed in the following form:

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

The rule priority is given in the first part, with lower x values indicating higher priorities. The second part holds the matching fields and associated actions. Exact and wildcard matches will be represented with the operators '$\equiv$' and '$\equiv_{pfx}$', respectively. Other notations used in the OF rules matching fields and action list are defined in Table II, together with a list of key notations.

B. Unicast Forwarding

The following couple of OF rules govern the unicast forwarding inside the datacenter d, for each VO $v \in V_\delta^d$:

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

[1] Based on Open vSwitch (http://openvswitch.org/) command-line syntax.

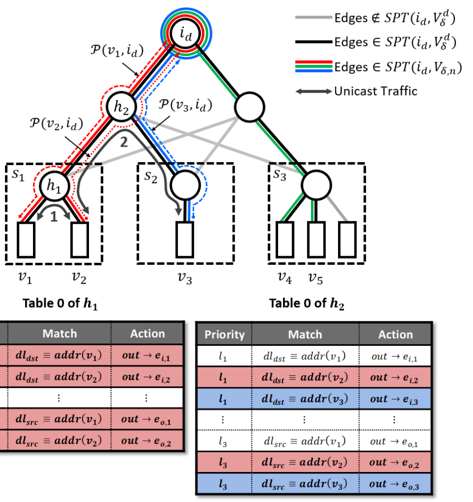


Table 0 of h_1

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 datacenter connectivity and unicast frame forwarding among VOs of three co-located clusters.

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

Both are installed $\forall h \in \mathcal{P}(v, i_d)$, $h \not\equiv i_d$, while only the rule (A) on i_d. With these, "precise" matching of v's L2 address with the destination and the source L2 addresses, respectively, can be achieved.

In more detail, if a frame generated by v is directed to another VO $\hat{v} \in V_{\delta,n}$, the frame matches both rules (A) and (B), i.e.:

$$\mathbf{p} \rightarrow l_1,\ \textbf{if}\ dl_{dst} \equiv addr(\hat{v}) \Rightarrow out \rightarrow e_i$$
$$\mathbf{p} \rightarrow l_3,\ \textbf{if}\ dl_{src} \equiv addr(v) \Rightarrow out \rightarrow e_o$$

in the first interconnection switch i^* where $\mathcal{P}(v, i_d)$ and $\mathcal{P}(\hat{v}, i_d)$ intersect. Such case is illustrated in Fig. 3 (i.e., on the switches h_1 and h_2, for the frames generated by v_2 for v_1 and v_3, respectively). While searching for i^*, rule (B) directs frames towards i_d, independently of their destination. Then, on i^*, rule (A) will be selected for its higher priority, consequently redirecting the frames towards the destination VO (i.e., v_1 and v_3 in the example). Furthermore, it can be noted that these rules allow direct communication between VOs of different clusters (i.e., v_2 and v_3 in Fig. 3).

When the destination VO is not in the same datacenter, frames will eventually reach i_d, and the forwarding behaviour will be driven by the wide-area algorithm thereon.

In the wide-area, the proposed algorithm relies on wildcard masks matching the Overlay and Center IDs (i.e., $id(\delta, c_n)$), rather than the precise matching of L2 addresses. This approach is particularly beneficial in the case where a significant number of VOs is clustered in each center.

The wide-area connectivity is realized through a "fully-meshed" overlay among N cluster centers. The algorithm works by obtaining the tree $Q_\delta^{c_n}$, $\forall c_n \in \{c_0, \dots, c_{N-1}\}$, according to Eq. (2). The following rule is installed:

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

This rule aims at directing unicast traffic from any other center c_m (mapped to $i_{d_m} \in Q_\delta^{c_n}$, $m \neq n$), towards the gateway switch i_{d_n} of the datacenter hosting the destination VO $v \in V_{\delta,n}$.

The wide-area unicast forwarding is designed to use the $Q_\delta^{c_n}$ tree rooted at the destination gateway switch to achieve

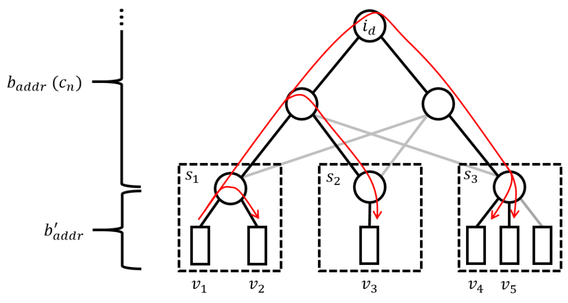


Fig. 4: Example of broadcast/multicast packet forwarding from v_1 to other VOs $v \in V_\delta$ in the same datacenter.

simpler rules, considering that there is only one path (and port) on the switches along $\mathcal{P}(i_{d_m}, i_{d_n})$, $m \neq n$, that is part of $Q_\delta^{c_n}$. Upon reaching i_{d_n}, the forwarding behaviour is again driven by the algorithm inside the datacenter.

C. Broadcast/Multicast Forwarding

Since broadcast and multicast forwarding rules are identical, apart from a difference in the matching MAC address, they will be treated as one in this paper. In order to guarantee isolation among the VTNs, IEEE 802.3 broadcast and multicast addresses are translated into overlay specific addresses by the first switch encountered by the packets (i.e., the one in the hypervisor).

Inside the datacenter d, the mechanism proposed relies on the same shortest-path tree $SPT(i_d, V_\delta^d)$, as shown in Fig. 4. A number of rules corresponding to the number of configured Q_δ cluster centers (N) is installed on every hop $h \in SPT(i_d, V_\delta^d)$.

If the hop $h \not\equiv i_d$ is not directly connected with any VO $v \in V_\delta^d$ (i.e., h does not correspond to the hypervisor switch in a server), the following type of forwarding rule is installed:

$$\mathbf{p} \rightarrow l_2,\ \textbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow out \rightarrow \mathcal{E} \tag{D}$$

where $\mathcal{E}$ is the set of all the edges at h belonging to $SPT(i_d, V_\delta^d)$. On $h \equiv i_d$, the forwarding behavior internally to the datacenter will correspond to the one in the rule (D). However, as i_d is part of both the datacenter and wide-area networks, a rule covering the two portions of the overlay is applied:

$$\mathbf{p} \rightarrow l_2,\ \textbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow out \rightarrow \breve{\mathcal{E}}_n \tag{E}$$

where $\breve{\mathcal{E}}_n$ is the set of edges at the gateway switch i_d, $d \in D_\delta : c_n \mapsto i_d$, with elements $e \in \{Q_\delta^{c_n} \cup SPT(i_d, V_\delta^d)\}$, hence considering both internal and wide-area connections.

In case $h \not\equiv i_d$ is directly connected with at least one VO $v \in V_\delta^d$, the rule should take into account the translation between the universal IEEE 802.3 broadcast address b'_{addr} and the ones used in the overlay $b_{addr}(c_n)$, $\forall n \in \{0, \ldots, N-1\}$. Particularly, let h be the hypervisor switch of the server $s \in \mathcal{S}_d$, and let the set $V_\delta^{d,s} \subseteq V_\delta^d$ comprise all VOs $v \in V_\delta$ residing in s. For each cluster center c_n, the following couple of rules is installed on h:

$$\mathbf{p} \rightarrow l_2,\ \textbf{if}\ dl_{dst} \equiv b'_{addr}\ \&\&\ p_{in} \equiv \mathcal{E}_n \Rightarrow \begin{bmatrix} dl_{dst} := b_{addr}(c_n),\ out \Rightarrow \mathcal{E} \cap \mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{N-1} \\ out \Rightarrow \mathcal{E}_0 \cup \ldots \cup \mathcal{E}_{N-1} \end{bmatrix} \tag{F}$$

$$\mathbf{p} \rightarrow l_2,\ \textbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow \begin{bmatrix} out \Rightarrow \mathcal{E} \cap \mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{N-1} \\ dl_{dst} := b'_{addr},\ out \Rightarrow \mathcal{E}_0 \cup \ldots \cup \mathcal{E}_{N-1} \end{bmatrix} \tag{G}$$

where $\mathcal{E}_n$ is the set of edges at h that directly connects to VOs bound to c_n, i.e.,

$$\mathcal{E}_n \triangleq \{e : e \in \mathcal{E} \wedge vertex(e) \cap V_{\delta,n} \neq \emptyset\} \tag{3}$$

As illustrated in Fig. 4, rule (F) handles broadcast packets originating from VOs in the datacenter and addressed to $b'_{addr} \equiv \mathbf{FF}:\mathbf{FF}:\mathbf{FF}:\mathbf{FF}:\mathbf{FF}:\mathbf{FF}$. In case of multiple VOs connected to the same hypervisor switch of the broadcast source, it will forward the broadcast frame to them without any change in the frame. Rule (F) is also in charge of translating b'_{addr} into one of the overlay broadcast addresses $b_{addr}(c_n)$, $n \in \{0, \ldots, N-1\}$, and of forwarding the modified frame to the rest of $SPT(i_d, V_\delta^d)$.

Conversely, when a frame arrives at switches directly connected to VOs (s_2 and s_3 in Fig. 4), the rule (G) is applied. Such rule will forward a copy of the unmodified frame onto links interconnecting further switches, and it will remap the broadcast address back to b'_{addr} in order to deliver the frame to all the VOs directly connected.

At the wide-area portion of the MCO network, rule (E) can also be applied to make broadcast traffic generated by VOs bound to c_n reach all the other hosts in Q_δ, as the set $\breve{\mathcal{E}}_n$ will only include edges $e \in Q_\delta^{c_n}$ on wide-area switches $i \in Q_\delta^{c_n}$, $i \not\equiv i_{d_n}$. In contrast to unicast forwarding, the $Q_\delta^{c_n}$ tree is crossed downstream from the root (which is the origin of broadcast traffic, and has a prefix $id(\delta, c_n)$) to the leaf nodes.

D. Fixed and Mobile Access Terminations

A network termination is defined to be a port of a switch in the network, where traffic transmitted/received by remote devices (in the home/enterprise network or connected through the radio access) is not carried on top of access carrier protocols/point-to-point tunnels. For this purpose, network terminations can be understood as the output ports of nodes performing the authentication and authorization on traffic coming from the wireline/radio access. Without loss of generality, the home and mobile access terminations of the users are supposed to be mapped on the Q_δ cluster centers $c_\hbar$, $c_m \in \{c_0, \ldots, c_{N-1}\}$, respectively.

It is worth noting that the proposed algorithm can support multiple mobile terminals, but only a single wireline/home termination. For the sake of simplicity, we will describe the single user case with home termination $\hbar_u$ and only a single mobile device, whose termination is represented by m_u.

In the case of the home termination, the algorithm has been specifically designed in order to support physical hosts in the domestic LAN without the need of translating their MAC addresses. This feature has been made possible by adding some further rules for: **(i)** making the traffic originating from/destined to the home LAN reach the center $c_\hbar$; and **(ii)** collecting the traffic addressed to devices in the home from all the other cluster centers in Q_δ.

Regarding the interconnectivity between $c_\hbar$ and $\hbar_u$, the shortest path $\mathcal{P}_\hbar$ is calculated, and for each hop $h \in \mathcal{P}_\hbar$, $h \not\equiv i_{d_\hbar}$ (i.e., $c_\hbar \mapsto i_{d_\hbar}$), the following rules are applied:

$$\mathbf{p} \rightarrow l_4,\ \textbf{if}\ dl_{src} \equiv_{pfx} id(\delta) \Rightarrow out \rightarrow e_i \tag{H}$$

$$\mathbf{p} \to l_4,\ \textbf{if}\ dl_{dst} \equiv_{pfx} id(\delta) \Rightarrow out \to e_o \qquad \text{(I)}$$

On $i_{d_\hbar}$, only rule (H) is configured, since the node already has rules of type (C) for reaching any VO bound to other centers in Q_δ. On the other hand, on the switch $i_\hbar$ hosting the termination $\hbar_u$ on the port $p_\hbar$, the following rule is applied instead of rule (I):

$$\mathbf{p} \to l_4,\ \textbf{if}\ dl_{dst} \equiv_{pfx} id(\delta)\ \&\&\ p_{in} \equiv p_\hbar \Rightarrow out \to e_o \quad \text{(J)}$$

In order to support broadcast traffic to/from the home termination $\hbar_u$, rule (D) is installed for each hop $h \in \mathcal{P}_\hbar$, $h \not\equiv i_\hbar$, and the following couple of rules on $i_\hbar$:

$$\mathbf{p} \to l_2,\ \textbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow dl_{dst} := b'_{addr},\ out \to e_i \qquad \text{(K)}$$

$$\mathbf{p} \to l_2,\ \textbf{if}\ dl_{dst} \equiv b'_{addr} \Rightarrow dl_{dst} := b_{addr}(c_n),\ out \to e_o \qquad \text{(L)}$$

Rules (K) and (L) are designed based on the assumption that $i_\hbar$ is a legacy switch that only hosts home terminations $\hbar_u$, and not VOs $v \in V_\delta^{d_\hbar}$.

Passing to the mobile connectivity, we refer to the 5G 3GPP Release 15 specifications [43]. In such a case, UEs could be exposed by the Network Exposure Function (NEF) and terminated by specific dedicated ports of User Plane Functions (UPFs), which shall run on the same datacenters hosting VOs. In this sense, we suppose that $\varphi(m_u) \equiv (i_m, p_m)$, where i_m is the switch connecting to the UPF instance serving user u's terminal and p_m is the port on i_m connected to m_u.

As in the previous cases, in order to connect m_u to the center c_m, the shortest path $\mathcal{P}_m$ among them is calculated and rules (A) and (B) are installed on each hop $h \in \mathcal{P}_m$, accordingly, except on the switch i_m, on which the following rules will be installed:

$$\mathbf{p} \to l_2,\ \textbf{if}\ dl_{dst} \equiv addr(m_u) \Rightarrow dl_{dst} := addr'(m_u),\ out \to p_m \qquad \text{(M)}$$

$$\mathbf{p} \to l_2,\ \textbf{if}\ p_{in} \equiv p_m \Rightarrow dl_{dst} := addr(m_u),\ out \to e_o \quad \text{(N)}$$

where $addr(\cdot)$ and $addr'(\cdot)$ are functions that return the overlay and physical L2 addresses, respectively.

Broadcast forwarding internal to the shortest-path tree $SPT(i_{d_m}, V_\delta^{d_m})$ is supported by the same rules inside the datacenter (i.e., rule (D) is installed on each hop $h \in \mathcal{P}_m$, $h \not\equiv i_{d_m}$, then the rules (F) and (G) on the switch i_m). On i_{d_m}, rule (E) is installed to cover both internal and wide-area interconnections.

V. Seamless Migration Support

The MCO is designed to efficiently support the seamless migration of VO(s), which might be decided by upper-lying orchestrators reacting upon specific events (e.g., UE handovers), or by planning the placement in advance (e.g., by means of probabilistic models identifying mobility patterns). When such reallocation is decided, it has to be performed by migrating the VO software instances, and by reconfiguring the network to maintain full connectivity before, during and after the software migration process. Only if these two operations are fulfilled with negligible service interruption times, the migration is considered to be "seamless". Although the software migration can be handled through various advanced software mechanisms (spanning from the copy of the entire execution container – i.e., migration of a VM or of a container – to the transfer of a limited set of status information – e.g., external database caches used by cloud-native software instances), network-induced performance drawbacks are still inevitable.

Legacy L2 Ethernet networking can incur further delays, due to reactive re-building of switching tables. SDN is well-known to easily overcome such drawbacks, and to enable seamless network reconfigurations during migrations in an effective fashion.

Common approaches for minimizing the service interruption time usually apply a two-step migration procedure [44], summarized as follows.

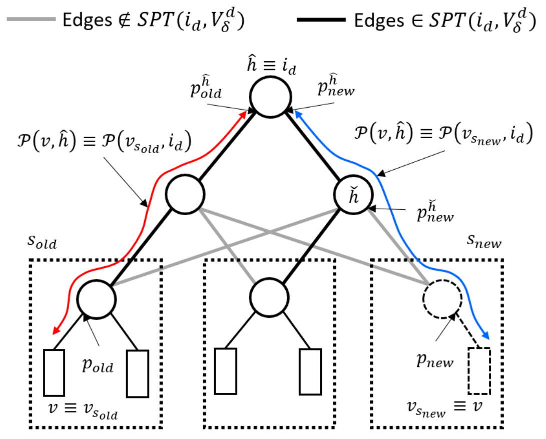
Step 1: Before initiating the migration, network switching/routing rules are temporarily configured in order to duplicate packets destined to the VO(s) on the move towards both "old" and "new" positions. Moreover, in order to guarantee the correct routing of the traffic generated from the new position(s) upon migration completion, also the new forwarding rules are calculated and configured on the involved nodes.
Step 2: Upon completion of the migration process, network switching/routing rules are updated in order to remove the connectivity to/from the old position(s), and maintain only the connectivity to/from the new position(s).

The MCO supports the aforementioned procedure for two types of migration: **(i)** *VO migration*: this only involves one VO, which is being migrated between two servers of the same in-network datacenter; and **(ii)** *center migration*: this consists of a bulk migration of all the VOs bound to a cluster center between two in-network datacenters. These two types have been designed to support reconfiguration of the underlying infrastructure and of overlying services, respectively.

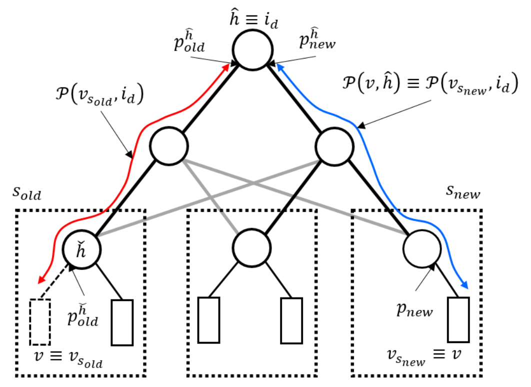
Particularly, the migration of single VOs between two servers of the same datacenter is a primitive operation mandatory to allow the maintenance of servers, or the dynamic consolidation of datacenter resources (e.g., for reducing the energy consumption by making idle servers entering standby modes). This is especially useful for operations at the infrastructure level, and it should be as transparent as possible to the service level (i.e., to the operation levels and performance provided by services running on VOs).

On the other hand, bulk migration of VOs has been defined to adapt the location of services in an efficient and scalable way. Through the migration of a VO cluster, it may be possible to reduce the end-to-end delay between the services running on such VOs and the end-user devices. For instance, if an end-user accesses services through his/her smartphone, center migrations can be useful to move clusters of VOs with similar QoS requirements closer to his/her mobile termination, also during hand-over from one network access point to another.

It can be noted that, in case the upper-lying orchestrators perform probabilistic placement of VO copies in the network, the MCO mechanism can support Step 1 operations to distribute the VO/cluster traffic towards multiple positions at the same time, and Step 2 operations to prune all the unnecessary endpoints.


Edges $\notin SPT(i_d, V_\delta^d)$
Edges $\in SPT(i_d, V_\delta^d)$
$\hat{h} \equiv i_d$
$p_{old}^{\hat{h}}$
$p_{new}^{\hat{h}}$
$\mathcal{P}(v,\hat{h}) \equiv \mathcal{P}(v_{s_{old}}, i_d)$
$\mathcal{P}(v,\hat{h}) \equiv \mathcal{P}(v_{s_{new}}, i_d)$
$\check{h}$
$p_{new}^{\check{h}}$
s_{old}
s_{new}
p_{old}
p_{new}
$v \equiv v_{s_{old}}$
$v_{s_{new}} \equiv v$

(a) before migration



(b) after migration

Fig. 5: Example of VO migration between two servers in the same in-network datacenter.

A. VO Migration

Let v be the VO to be migrated from the server s_{old} to s_{new}, as illustrated in Fig. 5. We suppose that $c_n \mapsto i_d$ is the cluster center to which v is bound, and $s_{old}, s_{new} \in \mathcal{S}_d$, $d \in D_\delta$. Moreover, p_{old} and p_{new} are the ports that connect v to the hypervisor switches in s_{old} and s_{new}, respectively.

As previously sketched, this type of migration only involves the overlay configuration inside a single datacenter, and no operations are needed at the backbone level. Therefore, Steps 1 and 2 of the migration procedure only concern the rules acting inside the datacenter, as detailed below.

S1: Let $\hat{h}$ be the switch where the paths $\mathcal{P}(v_{s_{old}}, i_d)$ and $\mathcal{P}(v_{s_{new}}, i_d)$ intersect, where v_s is the instance of v in the server $s \in \{s_{old}, s_{new}\}$. Since there are two paths (i.e., $\mathcal{P}(v,\hat{h}) \in \{\mathcal{P}(v_{s_{old}},\hat{h}), \mathcal{P}(v_{s_{new}},\hat{h})\}$) with the same matching rules for v, $\hat{h}$ will also have two edges towards v, $e_i \in \{p_{old}^{\hat{h}}, p_{new}^{\hat{h}}\}$, as shown in Fig. 5a – in effect, duplicating the traffic.

On $\hat{h}$, rule (A) is updated to (A′), by adding $p_{new}^{\hat{h}}$ as output interface – this allows unicast frames destined to v to be forwarded to both $p_{old}^{\hat{h}}$ and $p_{new}^{\hat{h}}$. Then, rules of the type (A) and (B) are configured along all the hops $h \in \mathcal{P}(v_{s_{new}}, \hat{h})$.

As regards broadcast traffic, we proceed in a similar fashion, by first identifying the switch $\check{h}$ where $\mathcal{P}(v_{s_{new}}, i_d)$ intersects the shortest-path tree $SPT(i_d, V_\delta^d)$. If $\check{h}$ does not coincide with the hypervisor switch of s_{new}: **(i)** the rule (D) (or rule (E), if $\check{h} \equiv i_d$) on $\check{h}$ is updated by adding $p_{new}^{\check{h}}$ to the output interfaces, where $p_{new}^{\check{h}}$ is the port on $\check{h}$ through which it is possible to reach v on s_{new}; **(ii)** rule (D) is added on each hop $h \in \mathcal{P}(s_{new}, \check{h})$; and **(iii)** rules (F) and (G) are added on s_{new}'s hypervisor switch. If $\check{h}$ coincides with the hypervisor switch of s_{new}, only rules (F) and (G) are updated by adding $p_{new}^{\check{h}}$ to the set $\mathcal{E}_n$ of edges at $\check{h}$ that directly connects to VOs that are bound to the center c_n.

S2: This step aims at pruning the unicast and broadcast connectivity towards the old position of v. For this purpose, the rule (A′) on $\hat{h}$ is updated back to (A) by removing the output port $p_{old}^{\hat{h}}$, and rules (A) and (B) are removed from all switches along $\mathcal{P}(v_{s_{old}}, \hat{h})$.

Now, let $\check{h}$ be the switch where $\mathcal{P}(v_{s_{old}}, i_d)$ intersects the shortest-path tree $SPT(i_d, V_\delta^d)$, as illustrated in Fig. 5b. If $\check{h}$ does not coincide with the hypervisor switch of s_{old}, the rule (D) (or (E), if $\check{h} \equiv i_d$) on $\check{h}$ is updated, by removing the port $p_{old}^{\check{h}}$ from the output interfaces. Moreover, rule (D) is removed from all switches along $\mathcal{P}(s_{old}, \check{h})$, as well as rules (F) and (G) from the hypervisor switch of s_{old}. On the contrary, if $\check{h}$ coincides with the hypervisor switch of s_{old}, only rules (F) and (G) are updated by removing $p_{old}^{\check{h}} \equiv p_{old}$ from the corresponding set $\mathcal{E}_n$ (defined by Eq. 3).

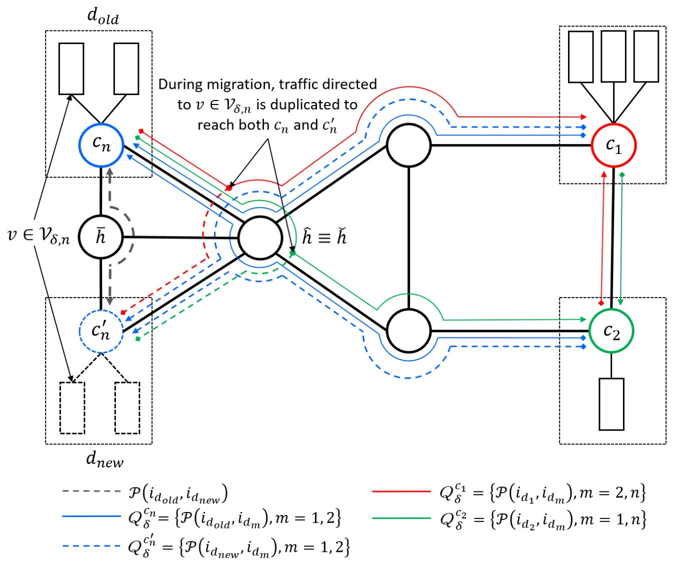

d_{old}
During migration, traffic directed to $v \in V_{\delta,n}$ is duplicated to reach both c_n and c_n'
c_n
c_1
$v \in V_{\delta,n}$
$\bar{h}$
$\hat{h} \equiv \check{h}$
c_n'
c_2
d_{new}
$\mathcal{P}(i_{d_{old}}, i_{d_{new}})$
$Q_\delta^{c_n} = \{\mathcal{P}(i_{d_{old}}, i_{d_m}), m = 1,2\}$
$Q_\delta^{c_n'} = \{\mathcal{P}(i_{d_{new}}, i_{d_m}), m = 1,2\}$
$Q_\delta^{c_1} = \{\mathcal{P}(i_{d_1}, i_{d_m}), m = 2,n\}$
$Q_\delta^{c_2} = \{\mathcal{P}(i_{d_2}, i_{d_m}), m = 1,n\}$

Fig. 6: Example of center migration between two datacenters.

B. Center Migration

Here we suppose that the VO cluster bound to the center c_n has to be migrated from the datacenter d_{old} to d_{new} (i.e., $c_n \mapsto i_{d_{old}}$ and $c_n' \mapsto i_{d_{new}}$), as depicted in Fig. 6. Unlike single VO migrations, Steps 1 and 2 in this type of migration concern the rules acting in both datacenter and wide-area domains, as detailed in the following.

S1: In this step, the datacenter and wide-area portions of the overlay are reconfigured for:

(i) bidirectionally propagating any frame exchanged among VOs bound to c_n (i.e., $v \in V_{\delta,n}$) between $i_{d_{old}}$ and $i_{d_{new}}$; **(ii)** delivering packets from the other centers c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, destined to c_n towards both datacenters d_{old} and d_{new}.

For this purpose, some rules of type (A) related to the VOs bound to c_n, in both d_{old} and d_{new}, are modified in order to make unicast frames destined to such VOs reach the datacenter gateway. Particularly, rule (A$''$) is derived from (A) by setting the output interfaces to ξ – this allows traffic to flow in either e_i or e_o directions of the path. Thus, rules of type (A$''$) are configured $\forall v \in V_{\delta,n}^{d^*} \cap V_{\delta,n}$, $d^* \in \{d_{old}, d_{new}\}$, and $\forall h \in \mathcal{P}(v, i_{d^*})$. On i_{d^*}, however, rules of type (A) are retained for forwarding of wide-area traffic destined to v. Rules of types (B), (D), (F) and (G) are also configured accordingly in d_{new}.

In order to guarantee the correct traffic exchange between VOs bound to c_n in d_{old} and d_{new}, the path $\mathcal{P}(c_n, c_n') \equiv \mathcal{P}(i_{d_{old}}, i_{d_{new}})$ in the wide-area portion is computed. Considering this path, the following rule is installed on $i_{d_{old}}$ and $i_{d_{new}}$:

$$\mathbf{p} \rightarrow l_1, \text{ if } dl_{dst} \equiv_{pfx} id(\delta, c_n) \; \&\& \; p_{in} \equiv p_{dc} \Rightarrow out \rightarrow \xi \tag{O}$$

and for each hop $h \in \mathcal{P}(c_n, c_n')$ (e.g., $\bar{h}$ as shown in the example in Fig. 6), the rule:

$$\mathbf{p} \rightarrow l_2, \text{ if } dl_{dst} \equiv_{pfx} id(\delta, c_n) \Rightarrow out \rightarrow \xi \tag{P}$$

In the former, specifying the input port as a datacenter connection (p_{dc}) avoids loops around this path.

On the other hand, the interconnection between the cluster center c_n' in $i_{d_{new}}$ towards/from any other center c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, is realized by identifying the intersection switch $\hat{h}$ between $\mathcal{P}(i_{d_{old}}, i_{d_m})$ and $\mathcal{P}(i_{d_{new}}, i_{d_m})$. On each hop $h \in \mathcal{P}(i_{d_{new}}, \hat{h})$, as well as on $i_{d_{new}}$, the rule (C) configured to match the prefix $id(\delta, c_m)$ is installed, allowing unicast frames from c_n' to reach c_m. In such a case, no updates are required on $\hat{h}$, since from $\hat{h}$ to any other center c_m the rules for the shortest-path trees $Q_\delta^{c_n}$ and $Q_\delta^{c_n'}$ are the same (see Fig. 6).

Conversely, to reach c_n' from c_m, the rule (C) configured to match the prefix $id(\delta, c_n)$ is installed on each hop $h \in \mathcal{P}(i_{d_{new}}, \hat{h})$, and is updated on $\hat{h}$ to (C$'$), by adding $p_{new}^{\hat{h}}$ as output interface, duplicating traffic towards the sub-paths $\mathcal{P}(i_{d_{old}}, \hat{h})$ and $\mathcal{P}(i_{d_{new}}, \hat{h})$, as illustrated in Fig. 6.

As regards broadcast traffic, in order to assure the correct delivery of frames generated by $v \in V_{\delta,n}$ in d_{old} towards d_{new}, and vice versa, the following rule is added to every hop $h \in \mathcal{P}(i_{d_{old}}, i_{d_{new}})$:

$$\mathbf{p} \rightarrow l_1, \text{ if } dl_{dst} \equiv b_{addr}(c_n) \Rightarrow out \rightarrow \xi \tag{Q}$$

Moreover, thanks to the rule (E) already present in $i_{d_{old}}$, broadcast frames generated by $v \in V_{\delta,n}$ in d_{new} are propagated to the other cluster centers through the shortest-path tree $Q_\delta^{c_n}$, whose root is in $i_{d_{old}}$.

Finally, rules of type (E) on wide-area switches are also installed/updated for delivering broadcast traffic generated in any other cluster center c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$. In particular, a new instance of rule (E) configured to match $b_{addr}(c_m)$ as destination address is installed on every hop $h \in \mathcal{P}(i_{d_{new}}, \check{h})$, where $\check{h}$ is the switch where $\mathcal{P}(i_{d_{new}}, i_{d_m})$ intersects the shortest-path tree $Q_\delta^{c_m}$. On $\check{h}$, the rule (E) matching $b_{addr}(c_m)$ is updated to (E$'$), by adding an output interface towards the sub-path $\mathcal{P}(\check{h}, i_{d_{new}})$.

S2: Upon the completion of all the migration operations $\forall v \in V_{\delta,n}$ from the datacenter d_{old} to d_{new}, this step is activated to prune all forwarding operations to/from d_{old} (at least for what concerns the traffic related to the center c_n).

To this end, rules of types (A) and (B) are removed $\forall v \in V_{\delta,n}$, and $\forall h \in \mathcal{P}(v, i_{d_{old}})$. Also, the shortest-path tree $SPT(i_{d_{old}}, V_\delta^{d_{old}})$ used for delivering broadcast messages inside d_{old} is modified accordingly, removing all the leaf nodes (i.e., corresponding to v, $\forall v \in V_{\delta,n}$) and related sub-paths to them. These operations are consequently reflected in the possible update and removal of the hops $h \in SPT(i_{d_{old}}, V_\delta^{d_{old}})$ of rules of type (D), and/or (F) and (G).

As regards the wide-area connectivity, the rule (C) matching $id(\delta, c_n)$ is removed $\forall h \in \mathcal{P}(i_{d_{old}}, \hat{h})$, while the rule (C$'$) on $\hat{h}$ is updated back to (C), by removing the port $p_{old}^{\hat{h}}$ from the output interfaces. The shortest-path trees $Q_\delta^{c_m}$, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, are updated in a similar fashion in the case where no other VO cluster is residing in d_{old}.

Finally, the shortest-path tree $Q_\delta^{c_n}$ is re-built, changing its root from $i_{d_{old}}$ to $i_{d_{new}}$. Let $Q_\delta^{c_n}$ and $Q_\delta^{c_n'}$ be the shortest-path trees calculated according to Eq. (2), with $i_{d_{old}}$ and $i_{d_{new}}$ as root nodes, respectively. The rule (E) matching $b_{addr}(c_n)$ is consequently: **(i)** removed from hop $\tilde{h}$, $\forall \tilde{h} : \tilde{h} \in Q_\delta^{c_n} \wedge \tilde{h} \notin Q_\delta^{c_n'}$; and **(ii)** added on hop $\tilde{h}$, $\forall \tilde{h} : \tilde{h} \notin Q_\delta^{c_n} \wedge \tilde{h} \in Q_\delta^{c_n'}$. Thanks to the optimal substructure property of the shortest paths composing $Q_\delta^{c_n}$ and $Q_\delta^{c_n'}$, the resulting spanning tree obtained with the above rule removals and additions will still be composed of the shortest paths among $i_{d_{new}}$ and any other cluster center c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$.

Note that this procedure cannot produce switching loops, since the forwarding rules deliver packets only towards the leaf nodes of $Q_\delta^{c_n}$ and $Q_\delta^{c_n'}$; hence, the possible and transitory presence of two root nodes could only produce duplicated broadcast packets. Moreover, since the same VO can reside in only one datacenter at a time, such duplicates can be avoided by removing rules of type (O), (P) and (Q), just before passing from $Q_\delta^{c_n}$ to $Q_\delta^{c_n'}$.

VI. Scalability and Performance Metrics

In this section, we introduce the set of metrics considered to evaluate MCO's scalability and performance, along with the different parameters used to understand their behaviour.

Regarding scalability, the key metrics that we considered are two: the number of OF rules needed to set up an MCO overlay, and the number of updates needed in case of VO/cluster migration. This choice was mainly driven by the fact that OF switches have finite-sized rule tables (usually in the order of some thousands [45]), and that the number updates/instantiations of OF rules directly affects the time needed by the SDN controller to apply the network reconfiguration to the involved switches (since each rule addition, deletion, or update has to be signalled through a separate OF message [9]).

As regards the performance, path lengths have already been evaluated in [15], demonstrating cases of path sub-optimality when the edges have random asymmetrical weights. In this work, the performance evaluation focuses on metrics related to the L2 overlay connectivity instantiation (i.e., rule

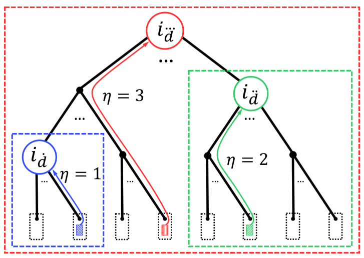


Fig. 7: Different datacenter depths $\eta \in \{1, 2, 3\}$.

calculation times) and center migration (i.e., rule update calculation times) processes that give indications on MCO's computational overhead.

A. Scalability

The number of forwarding rules depends on the number of VOs and switches involved in the overlay – and consequently, on the geographical distribution and topology of the datacenters, as well as the VO placement among them. For this purpose, we consider three *datacenter depths* to represent the upcoming Edge/Cloud interplay using the number of intermediate hops η between the VO and the datacenter gateway switch, as illustrated in Fig. 7. The traditional three-layer datacenter topology [46] of the Cloud ($\dddot{d}$) is supposed to be represented by the datacenter depth $\eta = 3$, while the heterogeneity among Edge nodes ($\dot{d}$ and $\ddot{d}$) by the depths $\eta \in \{1, 2\}$. This is particularly useful in counting unicast forwarding rules.

On the other hand, the scalability of handling bulk migrations between two datacenters is evaluated in terms of the number of rule updates in the wide-area portion of the overlay. This covers the rules installed/modified/removed on $i_{d_{old}}$, $i_{d_{new}}$, $\forall h \in \mathcal{P}(i_{d_{old}}, i_{d_{new}})$, and $\forall h \in \{\mathcal{P}(i_{d_{old}}, i_{d_m}), \mathcal{P}(i_{d_{new}}, i_{d_m})\}, \forall m \in \{0, \ldots, N-1\}, m \neq n$, in performing a center migration.

1) Number of forwarding rules: The rules for unicast (R_U) and broadcast (R_B) forwarding are counted separately in order to show their distinct behaviours – note that the sum of these two components indicates the total number of rules ($R = R_U + R_B$) needed to be installed for overlay connectivity.

Unicast R_U is simply the sum of the rules of types (A)–(C), and is given by:

$$\begin{aligned} R_U &= \sum_n \Big[\overbrace{|V_{\delta,n}|(\eta_{d_n} + 1)}^{\text{(A)}} + \overbrace{|V_{\delta,n}|(\eta_{d_n})}^{\text{(B)}} + \overbrace{(|Q_\delta^{c_n}| - 1)}^{\text{(C)}}\Big] \\ &= \sum_n \big[|V_{\delta,n}|(2\eta_{d_n} + 1) + |Q_\delta^{c_n}| - 1\big] \end{aligned} \quad (3)$$

where η_{d_n} is the number of intermediate hops from a VO $v \in V_{\delta,n}$ to the gateway switch i_{d_n}, $\forall n \in \{0, \ldots, N-1\}$.

Broadcast R_B, on the other hand, is obtained by summing up the rules of types (D)–(G), as follows, $\forall d \in D$:

$$\begin{aligned} R_B &= \sum_d \Big[\overbrace{N(|I_d^\delta| - 1)}^{\text{(D)}} + \overbrace{2N|\mathcal{S}_d^\delta|}^{\text{(F) and (G)}}\Big] + \sum_n \overbrace{|Q_\delta^{c_n}|}^{\text{(E)}} \\ &= \sum_d \big[N(|I_d^\delta| + 2|\mathcal{S}_d^\delta| - 1)\big] + \sum_n |Q_\delta^{c_n}| \end{aligned} \quad (4)$$

Considering that multiple cluster centers can be mapped to the same gateway switch, the *cluster/datacenter ratio* may vary over time – it would also be interesting to look at how the number of rules vary with this parameter, as we shall see later on.

2) Number of rule updates: For the sake of readability, we decompose the number of wide-area rule updates ($R_{M_{wa}}$) into two components (i.e., $R_{M_{wa}} = R^1_{M_{wa}} + R^2_{M_{wa}}$) that correspond to the two-step seamless migration procedure. Particularly, the number of rule updates in Step 1 is given by:

$$\begin{aligned} R^1_{M_{wa}} &= \overbrace{2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1)}^{\text{(O) – (Q)}} + \sum_m \Big[\overbrace{2(|\mathcal{P}(i_{d_{new}}, \hat{h}_m)| + 1)}^{\text{(C) and (C')}} \\ &\quad + \overbrace{(|\mathcal{P}(i_{d_{new}}, \check{h}_m)| + 1)}^{\text{(E) and (E')}}\Big] \\ &= 2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1) + \sum_m \big[2|\mathcal{P}(i_{d_{new}}, \hat{h}_m)| \\ &\quad + |\mathcal{P}(i_{d_{new}}, \check{h}_m)| + 3\big] \end{aligned} \quad (5)$$

where $\hat{h}_m$ (respectively, $\check{h}_m$), $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$ are the intersection switches between $\mathcal{P}(i_{d_{new}}, i_{d_m})$ and $\mathcal{P}(i_{d_{old}}, i_{d_m})$ (respectively, $Q_\delta^{c_m}$). A similar expression is obtained for the number of rule updates in Step 2:

$$\begin{aligned} R^2_{M_{wa}} &= \overbrace{2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1)}^{\text{(O) – (Q)}} + \sum_m \Big[\overbrace{2(|\mathcal{P}(i_{d_{old}}, \hat{h}_m)| + 1)}^{\text{(C) and (C')}} \\ &\quad + \overbrace{(|\mathcal{P}(i_{d_{old}}, \check{h}_m)| + 1)}^{\text{(E) and (E')}}\Big] \\ &= 2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1) + \sum_m \big[2|\mathcal{P}(i_{d_{old}}, \hat{h}_m)| \\ &\quad + |\mathcal{P}(i_{d_{old}}, \check{h}_m)| + 3\big] \end{aligned} \quad (6)$$

with $\check{h}_m$ now being the intersection switch between $\mathcal{P}(i_{d_{old}}, i_{d_m})$ and $Q_\delta^{c_m}$.

Furthermore, we define the *clustering index* as the number of VOs clustered in the center to be migrated, which is a parameter useful in studying the behaviour of this metric.

B. Performance

The performance metrics considered in this work are measurable from the timestamps generated by the MCO code. For this purpose, it is necessary to run the MCO algorithm over a given (emulated) Telecom infrastructure topology in order to collect measurements.

1) Rule calculation times: As users subscribe to new services, SCs are instantiated and added to their corresponding PNs. In this process, the time it takes to calculate the rules needed to be installed for the new SC gives indication on the computational complexity of the MCO algorithm. The behaviour of this metric is studied by considering different service chaining scenarios.

2) Rule update calculation times: Then, as users move around, center migration(s) may be initiated to meet the desired QoS/QoE. In this process, the time it takes to calculate the rule updates needed to perform a center migration gives indication on the time overhead incurred by the MCO algorithm in supporting seamless bulk migrations between datacenters.

VII. Numerical Results

MCO's scalability in terms of overlay implementation and bulk migration support is evaluated through a series of numerical simulations.

A. Simulation Framework

A simulation framework for a Telecom infrastructure with 30 datacenters interconnected by 100 transit nodes is implemented in Matlab. Datacenter depths $\eta \in \{1, 2, 3\}$ (see Section VI-A) are generated according to the probability mass function $f_\eta = \{0.6, 0.35, 0.05\}$, respectively. These parameters have been chosen by taking inspiration from state-of-the-art datacenter network architectures [46], as well as from the analyses done in [47]. Letting $\mathcal{T}$ be the set of access and interconnection switches, the logical interconnections $E_\mathcal{T}$ among the nodes $t \in \mathcal{T}$ are then generated randomly to form a graph-based topology $G(\mathcal{T}, E_\mathcal{T})$. Finally, based on the resulting topology, a transit node $t \in \mathcal{T}$ is randomly chosen for each $d \in D$, with constraint on the minimum number of hops (H_{min}) between any pair of datacenters $(d_a, d_b) \in D$, whose value highly depends on the graph size and topology. In this work, $H_{min} \leq 3$ is required in order to obtain a solution, and the value $H_{min} = 3$ is used to maximize the topological distribution of datacenters.

20 runs with varying seeds are executed for each parameter configuration to show the 95% confidence intervals in the results, with each run corresponding to a unique infrastructure topology.

B. Overlay Implementation

For unicast forwarding, the proposed approach (**MCO**) is evaluated in comparison with three baselines – $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$. $\mathbf{B_1}$ and $\mathbf{B_3}$ basically correspond to the *fully-meshed* and *OpenStack* cases considered in [15], respectively. In more detail, $\mathbf{B_1}$ corresponds to installing a couple of exact matching rules for each source-destination VOs in all the crossed switches in the datacenter and wide-area networks; this implies a "fully-meshed" OF connectivity among VOs, since each involved switch will have rules for exactly matching the MAC addresses of all VOs in the overlay. $\mathbf{B_2}$ is equivalent to the previous case, but the flow table compression proposed in [45] has been applied. The OpenStack baseline, $\mathbf{B_3}$, corresponds to the scenario where only the hypervisor switches are OF-enabled. In this case, the hypervisor switches communicate through tunnels, rather than by means of explicit OF rules on other interconnection switches. Thus, in $\mathbf{B_3}$, OF rules are only hosted in hypervisor switches, and are designed to work with exact matching "fully-meshed" connectivity (i.e., each hypervisor switch of the servers hosting a VO implements explicit rules to/from any other VOs in the overlay). It is important to note that conventional tunneling protocols used on top of the OpenStack platform [48] also incur additional costs.

As regards broadcast forwarding, the baseline $\mathbf{B_{1/2/3}}$ corresponds to the case with a single overlay broadcast address; hence, only one rule is installed on the switches involved in Q_δ – except on the hypervisor switches, where two rules are installed for the mapping between the overlay and universal broadcast addresses. The wide-area connectivity is determined by the minimum spanning tree $Q_\delta^{i_d *} : |Q_\delta^{i_d *}| = min_{d \in D_\delta} |Q_\delta^{i_d}|$. Adding the values obtained in $\mathbf{B_{1/2/3}}$ to those of $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$ results in the total number of forwarding rules in the considered baselines. A summary of the aforementioned baselines is reported in Table III, for quick reference.

TABLE III: Overlay implementation baselines.

Baseline	Description
$\mathbf{B_1}$	fully-meshed: installs exact matching rules for each source/destination pair of VOs on all switches involved
$\mathbf{B_2}$	two-level flow aggregation: compression of rules to combine flows towards the same server/datacenter
$\mathbf{B_3}$	OpenStack: installs exact matching rules for each source/destination pair of VOs only on hypervisor switches of servers involved
$\mathbf{B_{1/2/3}}$	single overlay broadcast address, wide-area connectivity determined by the minimum spanning tree

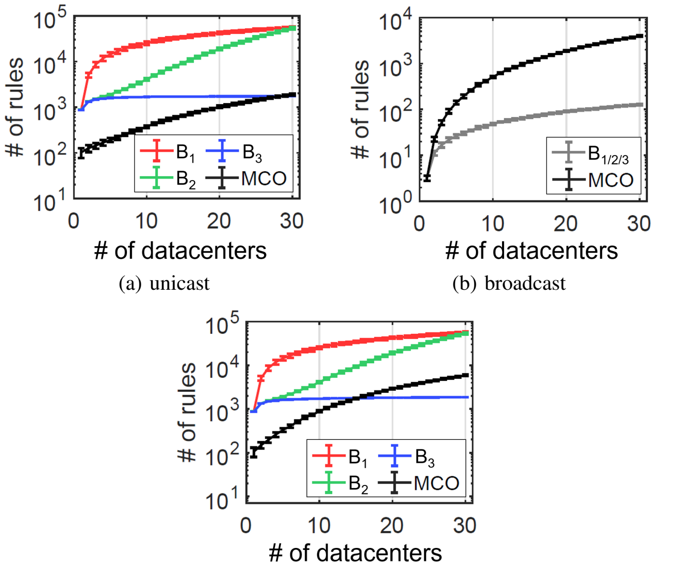
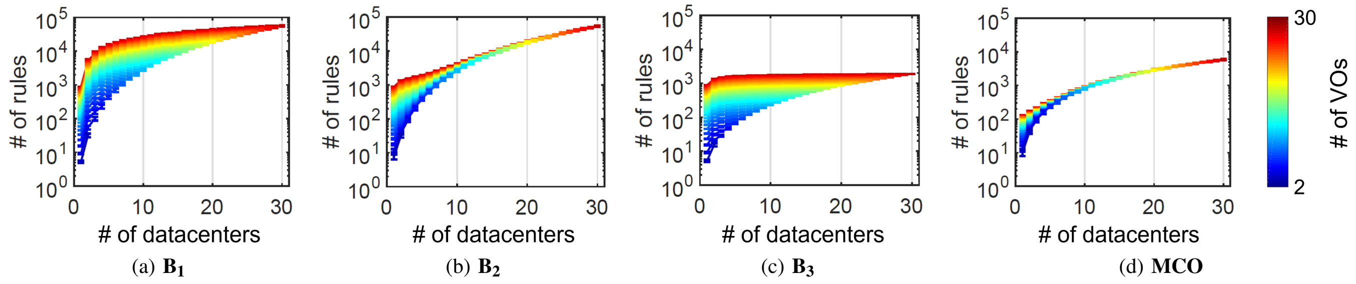


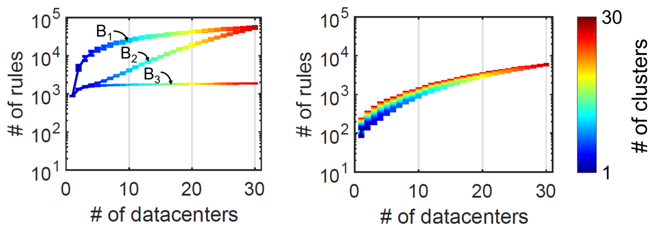
Fig. 8: Number of forwarding rules in the **MCO**, $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$ cases, given 30 VOs.

Suppose that there is one-to-one correspondence between clusters and datacenters, and VOs are uniformly distributed among N datacenters. With 30 VOs in V_δ, all communicating with each other, Fig. 8a shows that **MCO** has up to over $1 \sim 2$ order of magnitude less rules than the three baselines, depending on the number of datacenters involved. On the other hand, **MCO** has more broadcast rules than $\mathbf{B_{1/2/3}}$, as shown in Fig. 8b, stemming from the N overlay broadcast addresses and wide-area trees involved in broadcast forwarding. Despite this, the total number of forwarding rules of **MCO** remains better than $\mathbf{B_1}$ and $\mathbf{B_2}$ by a considerable difference, as well as for small values of N in $\mathbf{B_3}$ (i.e., at $N \approx 15$, **MCO** starts to have more rules than $\mathbf{B_3}$), as shown in Fig. 8c. Moreover, the N wide-area trees rooted at each datacenter gateway involved are expected to provide better paths based on where broadcast traffic is generated, compared to those given only by $Q_\delta^{i_d *}$.



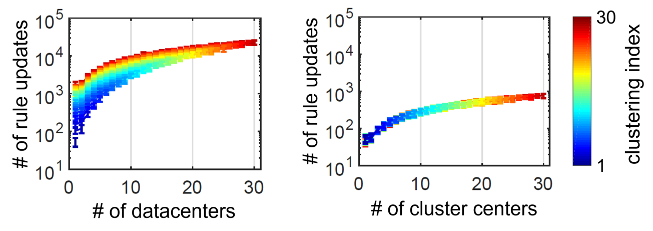
(a) $\mathbf{B_1}$ (b) $\mathbf{B_2}$ (c) $\mathbf{B_3}$ (d) **MCO**

Fig. 9: Impact of the number of VOs in an MCO network on the number of forwarding rules.



(a) $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$ (b) **MCO**

Fig. 10: Impact of the cluster/datacenter ratio on the number of forwarding rules, given 30 VOs.



(a) $\mathbf{B_M}$ (b) **MCO**

Fig. 11: Impact of the clustering index on the number of rule updates during bulk migrations.

1) Impact of the number of VOs: The impact of the number of VOs in V_δ on the number of forwarding rules in **MCO** and the three baselines is illustrated in Fig. 9. A stronger dependence on the number of VOs can be observed for $\mathbf{B_1}$ and $\mathbf{B_3}$ due to the exact matching rules installed for each source/destination pair of VOs in unicast forwarding. By counting only the unicast rules on the hypervisor switches plus the baseline broadcast rules, the latter also displays a stabilizing behaviour with increasing number of datacenters. Conversely, the advantage of flow aggregation in $\mathbf{B_2}$ becomes more evident as the number of datacenters involved increases, while **MCO** exhibits the least dependence on the number of VOs with its particular forwarding algorithm inside and among the datacenters.

2) Impact of the cluster/datacenter ratio: Recall that multiple cluster centers can be mapped to the same datacenter gateway over time – in such a case, the initial assumption of one-to-one correspondence between clusters and datacenters is no longer true.

With this in mind, Fig. 10 illustrates the impact of the cluster/datacenter ratio on the number of forwarding rules in **MCO** and the three baselines. As expected, $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$ are only dependent on the number of datacenters involved, while for **MCO** the number of forwarding rules increases with the cluster/datacenter ratio. This behaviour of the **MCO** presents a trade-off between scalability and flexibility. Particularly, in handling bulk migrations, flexibility is improved as VOs in V_δ^d with similar QoS/QoE requirements are clustered together – possibly, into multiple centers that can be migrated independently.

C. Seamless Mobility Support

For the wide-area rule updates during seamless bulk migrations, **MCO**'s center migration approach is evaluated with the baseline $\mathbf{B_M}$, which corresponds to the case of multiple VO migrations. Fig. 11 illustrates how the number of rule updates vary with the clustering index and the number of datacenters/clusters involved for the two cases, respectively. With a migration initiated for each VO, $\mathbf{B_M}$ shows a stronger dependence on the clustering index, while **MCO** is basically agnostic to the parameter and only depends on the number of clusters involved. Consequently, **MCO** results in up to over one order of magnitude less number of rule updates than $\mathbf{B_M}$.

VIII. Experimental Results

While MCO has multiple possible applications (e.g., mass-scale services), we evaluate its performance by considering the INPUT use case of *"PN-as-a-service" (PNaaS)* [49] [50]. Particularly, a PN is a special type of VTN that interfaces the user's private network with the VTN(s) of service providers (referred to as BNs henceforth). In this section, details on the use case, experimental testbed and considered SC scenarios are presented together with the obtained results.

A. The INPUT Use Case

The INPUT framework seeks to provide seamless experiences to its (mobile) users by guaranteeing a certain level of proximity to the VOs involved – not only in their PNs, but also in the BNs; PN-BN interactions highly depend on users' subscription to services. To do so, it leverages VO portability in the Edge Computing environment, enabling services to "follow" users, as needed. In detail, each PN or BN is an MCO network, where VOs are organized into clusters based on the required *proximity level* $p \in \{\mathbf{1}, \ldots, \mathbf{P}\}$, with lower values of p indicating tighter requirements.

Fig. 12 illustrates an example of a PN that interacts with 2 BNs. As previously anticipated, some VOs (i.e., v_2 and v_3) in the PN are also associated to cluster centers of BNs (i.e., $\mathbf{BN_1}$ and $\mathbf{BN_2}$, respectively). Since a VO v can only reside on a single datacenter, it follows that the PN and BN(s) centers to which v is bound to must have the same proximity level, and be mapped on the same datacenter. Therefore, when a PN

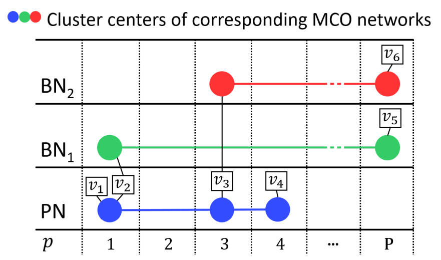


Fig. 12: Clustering VOs based on the required proximity level p and network domain (PN/BNs).

center with proximity level p is migrated, the corresponding center(s) in the BN(s) is/are also migrated.

The MCO algorithm is currently implemented in the *crater* module of the **OpenVolcano** platform [51] for the INPUT project testbed.

B. Experimental Testbed

To evaluate MCO's performance, the **OpenVolcano** platform (which is running on a Linux server equipped with an Intel® Xeon® E5-2620 v4 2.10GHz processor) is used to emulate an underlying Telecom infrastructure, starting from a real wide-area topology (namely, the **Interroute** topology, obtained from the datasets available in [52]).

In more detail, we adapt the **Interroute** topology to consist of 20 in-network datacenter nodes (i.e, randomly selected, and as before, with $H_{min} = 3$ as constraint in order to obtain a solution that maximizes the topological distribution of datacenters) and 90 transit nodes, interconnected by 148 edges (i.e., after removing self-loops). Contrary to the simulation framework, we consider three *datacenter sizes* in the experiments, rather than depths, to indicate the number of servers in each Edge Computing facility (i.e., small (20), medium (50) and large (100)). The size is chosen based on the number of wide-area edges a datacenter has: 2, 3 and > 3, respectively, under the assumption that large datacenters are more central and well connected than small and medium-sized ones. For simplicity, but without loss of generality, interconnection switches between the datacenter gateways and the servers are not considered in the emulation.

To add statistical significance in the results, each test is repeated 100 times with random CPU and RAM capacities/requirements among servers/VOs, introducing variations in the VO placement inside a datacenter (i.e., based on **OpenVolcano**'s placement policy, which is currently on a testing phase and beyond the scope of this work), from which the 95% confidence intervals are obtained. Nonetheless, any VO placement/consolidation policy can be applied.

TABLE IV: MCO parameters for the considered SC scenarios.

SC	# of BNs	# of VOs (PN + BNs)	# of Clusters PN	# of Clusters BNs	# of VOs/cluster PN	# of VOs/cluster BNs
SC$_1$	5	15	5	10	2	2
SC$_2$	10	30	10	19	2	2
SC$_3$	5	60	5	10	11	2
SC$_4$	5	15	5	30	2	1
SC$_5$	10	30	10	100	2	$1 \sim 2$
SC$_6$	5	60	5	30	11	1

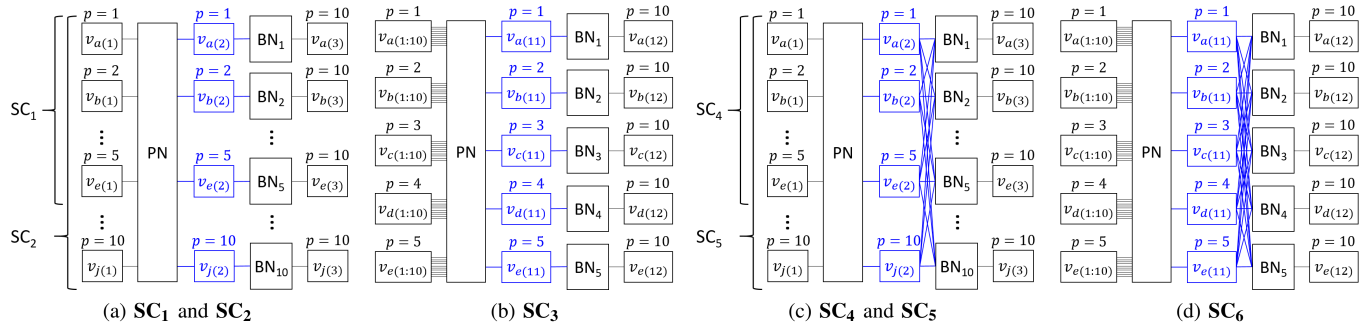
C. SC Instantiation Complexity

Six SC scenarios (i.e., **SC$_1$**,..., **SC$_6$**) are considered in this work. As illustrated in Fig. 13 and summarized in Table IV, these SCs are designed to cover variations in the number of BNs (and their interaction with the PN), clusters (PN + BNs) and VOs involved, while keeping modularity to easily automate their generation in the experiments.

Looking at Fig. 13, the VOs highlighted in blue (i.e., $v_{a(2)}, \ldots, v_{j(2)}$ or $v_{a(11)}, \ldots, v_{e(11)}$) are the ones that are identified with two or more cluster centers (i.e., one in the PN and another one in each BN connected); recall that all centers associated to the same VO must have the same p value and be mapped on the same datacenter.

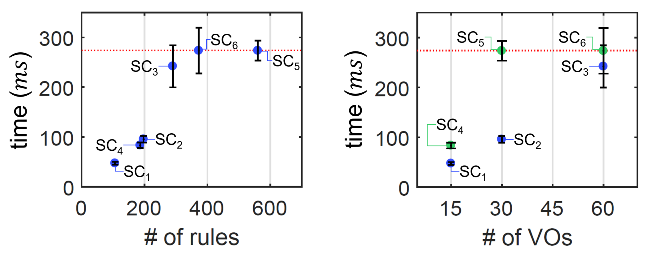
Note that the time required to deploy SCs highly depends on the instantiation of their respective L2 overlay connectivity. With MCO, the rules required to instantiate the connectivity for either of the six SCs can be calculated in less than 350 ms, as shown in Fig. 14; this demonstrates MCO's low computational complexity, even for more intricate SC scenarios (e.g., **SC$_5$** and **SC$_6$**).

It is interesting to note that while the number of rules increases with SC complexity, the average rule calculation time approaches a limit of around 273 ms, as indicated by the red dotted lines. In Fig. 14a, this sublinear behaviour manifests the impact of PN-BN interactions, mainly driven by both the number of VOs and VTNs involved – e.g., **SC$_3$** and **SC$_6$** correspond to the highest number of VOs (at least twice than other SCs), while **SC$_5$** to the highest number of BNs (over three times than other SCs). In Fig. 14b, a similar relationship is also observed between the measured times and



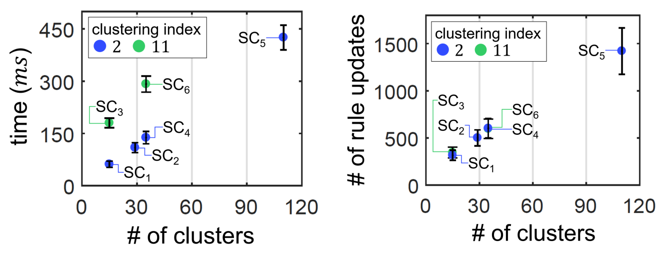
(a) **SC$_1$** and **SC$_2$** (b) **SC$_3$** (c) **SC$_4$** and **SC$_5$** (d) **SC$_6$**

Fig. 13: SC scenarios.



(a) vs the # of rules (b) vs the # of VOs

Fig. 14: Rule calculation times for SC instantiation.



(a) calculation times (b) # of rule updates

Fig. 15: Rule updates and calculation times during a center migration, for varying number of clusters involved (PN + BNs).

the number of VOs involved, with the slope determined by the PN-BN interactions, until the said limit is reached. Wider confidence intervals are observed for SCs involving more VOs (i.e., $\mathbf{SC_3}$ and $\mathbf{SC_6}$) since the VO clusters are more likely to be distributed to different number of servers in each test.

D. Center Migration Overhead

Considering the same SCs (which also cover variations in the clustering index), we emulate a center migration for c_1 (i.e, $p = 1$) of the PN to various destination datacenters in the Telecom infrastructure, with distances between d_{old} and d_{new} ranging from 8 to 12 hops. As previously mentioned, the corresponding cluster center(s) (i.e, $p = 1$) in the involved BN(s) will also be migrated to d_{new}.

Note that a migration can only begin when the necessary updates in the L2 overlay connectivity are already in place. The rule updates required for the center migration in either of the six SCs can be calculated in less than 500 ms, as illustrated in Fig. 15a; this demonstrates that the time overhead incurred by MCO's center migration approach for seamless bulk migration is practically negligible with respect to the total migration duration (e.g., tens to hundreds of seconds [50], and highly depends on the virtualization/migration technologies used, workloads and migration paths [53]–[55]).

It can be observed in Fig. 15b that the number of rule updates increases linearly with the number of clusters involved (PN + BNs), almost independently of c_1's clustering index; the latter only impacts the number of rule updates inside the datacenters, which is expected of MCO's center migration approach. A wider confidence interval is observed for $\mathbf{SC_5}$ since migrating the PN's center c_1 also migrates the corresponding centers (i.e, $p = 1$) of BNs $\mathbf{BN_1}$ through $\mathbf{BN_{10}}$, resulting in a multiplicative effect in the variation of the number of rule updates for different distances between d_{old} and d_{new}.

A similar linear behaviour is also observed for the calculation times, except that the $y-$intercept is determined by the clustering index. Particularly, the increase in the calculation times with increasing clustering index can be attributed to the (re-)running of the shortest-path algorithm for each of the VOs being migrated, in both d_{old} and d_{new}; still, the increase in time is not linearly proportional to the increase in the clustering index.

IX. Conclusions

Supporting both heterogeneity and mobility of Edge applications entails customized and dynamic traffic handling over a shared Telecom infrastructure. While SDN provides major gains in this respect, scalability is still an open problem. Considering an SDN-enabled environment, this paper proposed the MCO mechanism that offers high scalability in realizing wide-area VTNs and seamless mobility of VOs, among other aspects.

In contrast to state-of-the-art SDN mechanisms, numerical results show that MCO achieves up to over one order of magnitude smaller number of OF rules in the overlay implementation (compared to the fully-meshed, two-level flow aggregation and OpenStack cases), and rule updates during center migrations (compared to initiating multiple VO migrations), demonstrating its high scalability. MCO's performance has been also experimentally evaluated using the **OpenVolcano** platform in the context of the INPUT use case (i.e., PNaaS). Experimental results demonstrate its low computational complexity with < 350 ms rule calculation times for SC instantiation, even with more intricate SC scenarios. Within such complex PN-BN environments, a very low time overhead (i.e., < 500 ms rule update calculation times) is also observed during center migrations.

Acknowledgment

This work was partially supported by the INPUT and MATILDA projects, funded by the European Commission under the Horizon 2020 Programme (Grants no. 644672 and and no. 761898, respectively).

References

[1] B. Kim and B. Lee, "Integrated Management System for Distributed Micro-Datacenters," in *Proc. 18th Int. Conf. Adv. Commun. Technol. (ICACT)*, PyeongChang, Korea, Jan. 2016, pp. 466–469.

[2] M. Yannuzzi et al., "A New Era for Cities with Fog Computing," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 54–67, Mar. 2017.

[3] D. Sabella et al., "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] D. Taibi et al., "Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation," *IEEE Cloud Comput.*, vol. 4, no. 5, pp. 22–32, Sep. 2017.

[5] D. Bhamare et al., "A Survey on Service Function Chaining," *J. Netw. Comput. Appl.*, vol. 75, pp. 138–15, Nov. 2016.

[6] "LTE; Telecommunication Management; Life Cycle Management (LCM) for Mobile Networks that Include Virtualized Network Functions; Requirements," ETSI NFV Tech. Spec., Jul. 2017. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/128500_128599/128525/14.01.00_60-/ts_128525v140100p.pdf

[7] M. Nitti et al., "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.

[8] B. A. A. Nunes et al., "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.

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

[10] H. Wang et al., "Virtual Machine Migration Planning in Software-Defined Networks," *IEEE Trans. Cloud Comput. (Early Access)*, 2017.

[11] C. Clark et al., "Live Migration of Virtual Machines," in *Proc. 2nd USENIX Symp. Netw. Sys. Design Impl. (NSDI)*, vol. 2, Boston, MA, USA, May 2005, pp. 273–286.

[12] M. Mahalingam et al., "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," IETF RFC 7348, Aug. 2014. [Online]. Available: https://tools.ietf.org/html/rfc7348

[13] R. Niranjan Mysore et al., "PortLand: A Scalable Fault-tolerant Layer 2 Data Center Network Fabric," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, Oct. 2009.

[14] R. Kawashima and H. Matsuo, "Non-Tunneling Edge-Overlay Model using OpenFlow for Cloud Datacenter Networks," in *Proc. 5th IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Bristol, UK, Dec. 2013, pp. 176–181.

[15] R. Bruschi et al., "A Scalable SDN Slicing Scheme for Multi-domain Fog/Cloud Services," in *Proc. 2017 IEEE Conf. Netw. Softwarization (NetSoft)*, Bologna, Italy, Jul. 2017.

[16] "The INPUT Project." [Online]. Available: http://www.input-project.eu/

[17] A. Schwabe and K. Holger, "Using MAC Addresses as Efficient Routing Labels in Data Centers," in *Proc. 3rd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, Chicago, IL, USA, Aug. 2014, pp. 115–120.

[18] A. Hari et al., "Path Switching: Reduced-State Flow Handling in SDN Using Path Information," in *Proc. 11th ACM Int. Conf. Emerg. Netw. Experiments Technol. (CoNEXT)*, Heidelberg, Germany, Dec. 2015.

[19] K. Agarwal et al., "Shadow MACs: Scalable Label-switching for Commodity Ethernet," in *Proc. 3rd Workshop Hot Topics Soft. Defined Netw. (HotSDN)*, Chicago, IL, USA, Aug. 2014, pp. 157–162.

[20] S. Hu et al., "Explicit Path Control in Commodity Data Centers: Design and Applications," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2768–2781, Oct. 2016.

[21] O. Yazir et al., "Dynamic Resource Allocation in Computing Clouds Using Distributed Multiple Criteria Decision Analysis," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. (CLOUD)*, Miami, FL, USA, Jul. 2010, pp. 91–98.

[22] H. Goudarzi and M. Pedram, "Multi-dimensional SLA-Based Resource Allocation for Multi-tier Cloud Computing Systems," in *Proc. IEEE 4th Int. Conf. Cloud Comput. (CLOUD)*, Washington, DC, USA, Jul. 2011, pp. 324–331.

[23] K. Mills et al., "Comparing VM-placement Algorithms for On-demand Clouds," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Athens, Greece, Dec. 2011, pp. 91–98.

[24] Y. Awano and S. Kuribayashi, "A Joint Multiple Resource Allocation Method for Cloud Computing Environments with Different QoS to Users at Multiple Locations," in *Proc. IEEE Pacific Rim Conf. Commun. Comput. Signal Process. (PACRIM)*, Victoria, Canada, Aug. 2013, pp. 1–5.

[25] F. Pires and B. Baran, "Multi-objective Virtual Machine Placement with Service Level Agreement: A Memetic Algorithm Approach," in *Proc. IEEE/ACM 6th Int. Conf. Utility Cloud Comput. (UCC)*, Dresden, Germany, Dec. 2013, pp. 203–210.

[26] M. Sharkh et al., "Resource Allocation in a Network-based Cloud Computing Environment: Design Challenges," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 46–52, Nov. 2013.

[27] K. Su et al., "Affinity and Conflict-Aware Placement of Virtual Machines in Heterogeneous Data Centers," in *Proc. IEEE 12th Int. Symp. Auton. Decentralized Syst. (ISADS)*, Taichung, Taiwan, Mar. 2015, pp. 289–294.

[28] M. Bagaa et al., "Service-Aware Network Function Placement for Efficient Traffic Handling in Carrier Cloud," in *Proc. 2014 IEEE Wireless Commun. Netw. Conf. (WCNC)*, Istanbul, Turkey, Apr. 2014, pp. 2402–2407.

[29] R. Cohen et al., "Near-optimal Placement of Virtual Network Functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, Apr. 2015, pp. 1346–1354.

[30] D. Bhamare et al., "Multi-Cloud Distribution of Virtual Functions and Dynamic Service Deployment: OpenADN Perspective," in *Proc. 2015 IEEE Int. Conf. Cloud Eng.*, Tempe, AZ, USA, Mar. 2015, pp. 299–304.

[31] T. Taleb et al., "User Mobility-Aware Virtual Network Function Placement for Virtual 5G Network Infrastructure," in *Proc. 2015 IEEE Int. Conf. Commun. (ICC)*, London, UK, Jun. 2015, pp. 3879–3884.

[32] D. Bhamare et al., "Optimal Virtual Network Function Placement and Resource Allocation in Multi-Cloud Service Function Chaining Architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017.

[33] ——, "Multi-objective Scheduling of Micro-services for Optimal Service Function Chains," in *Proc. 2017 IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[34] X. Li et al., "Network Slicing for 5G: Challenges and Opportunities," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 20–27, Sep. 2017.

[35] A. Laghrissi et al., "Towards Edge Slicing: VNF Placement Algorithms for a Dynamic and Realistic Edge Cloud Environment," in *Proc. 2017 IEEE Global Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.

[36] D. Bhamare et al., "Efficient Virtual Network Function Placement Strategies for Cloud Radio Access Networks," *Comput. Commun.*, vol. 127, pp. 50–60, 2018.

[37] M. Bagaa et al., "Efficient Virtual Evolved Packet Core Deployment Across Multiple Cloud Domains," in *Proc. 2018 IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, Apr. 2018.

[38] J. Prados et al., "A Queuing based Dynamic Auto Scaling Algorithm for the LTE EPC Control Plane," in *Proc. 2018 IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, Dec. 2018.

[39] R. A. Addad et al., "Towards Modeling Cross-Domain Network Slices for 5G," in *Proc. 2018 IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, Dec. 2018.

[40] S. Bakiras, "Approximate Server Selection Algorithms in Content Distribution Networks," in *Proc. 2005 IEEE Int. Conf. Commun. (ICC)*, Seoul, South Korea, May 2005, pp. 1490–1494.

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

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

[43] The 3GPP Association, "System Architecture for the 5G System," 3GPP TS 23.501, Stage 2, Rel. 15, v.15.2.0, Jun. 2018.

[44] S. Ghorbani et al., "Transparent, Live Migration of a Software-Defined Network," in *Proc. 2014 ACM Symp. Cloud Comput. (SOCC)*, Seattle, WA, USA, Nov. 2014, pp. 3:1–3:14.

[45] B. Leng et al., "FTRS: A Mechanism For Reducing Flow Table Entries in Software Defined Networks," *Comput. Netw.*, vol. 122, pp. 1–15, Jul. 2017.

[46] M. F. Bari et al., "Data Center Network Virtualization: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, Q2 2013.

[47] R. Bruschi et al., "Evaluating the Impact of Micro-Data Center (μDC) Placement in an Urban Environment," in *Proc. 2018 IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Verona, Italy, Nov. 2018.

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

[49] R. Bruschi et al., "OpenStack Extension for Fog-Powered Personal Services Deployment," in *Proc. 29th Int. Teletraffic Congr. (ITC)*, vol. 2, Genoa, Italy, Sep. 2017, pp. 19–23.

[50] ——, "An SDN/NFV Platform for Personal Cloud Services," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 1143–1156, Dec. 2017.

[51] "OpenVolcano – Open Virtualization Operating Layer for Cloud/fog Advanced NetwOrks." [Online]. Available: http://openvolcano.org

[52] "The Internet Topology Zoo." [Online]. Available: http://topology-zoo.org

[53] D. Williams et al., "VirtualWires for Live Migrating Virtual Networks Across Clouds," IBM Res. Rep., Apr. 2013. [Online]. Available: http://domino.research.ibm.com/library/cyberdig.nsf/papers/FD9A14E5-9B138E7E85257B6000572CC3/$File/rc25378.pdf

[54] T. Wood et al., "CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1568–1583, Oct. 2015.

[55] W. Cerroni and F. Esposito, "Optimizing Live Migration of Multiple Virtual Machines," *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 1096–1109, Oct. 2018.