Towards Cross-Domain Service Function Chain Orchestration

Nassima Toumi[1,2], Olivier Bernier[1], Djamal-Eddine Meddour[1], Adlen Ksentini[2]
[1] Orange Labs Networks, Lannion, France
[2] EURECOM, Sophia-Antipolis, France
Email: {nassima.toumi, olivier.bernier, djamal.meddour}@orange.com, adlen.ksentini@eurecom.fr

***Abstract*—Service Function Chaining (SFC) refers to the process of steering packets between a set of functions to deliver an end-to-end service. It is considered as one of the enabling technologies for 5G along with Software Defined Networks (SDN) and Network Function Virtualization (NFV). One of the challenges for SFC deployment is the end-to-end orchestration, particularly in a multi-domain scenario where additional issues, such as the lack of visibility and control, and interoperability, need to be taken into account. In this paper, we propose a novel framework that leverages on and extends existing standards in order to perform an end-to-end cross-domain orchestration of SFCs, and ensure the desired forwarding of packets between the domains. A Proof of Concept implementation of our approach is also deployed and evaluated.**

I. Introduction

New generations of networks, including 5G, rely on a set of network softwarization technologies and paradigms in order to accommodate new emerging use cases: (i) Network Function Virtualization (NFV) that allows an improved flexibility in management compared to traditional networks; (ii) Software Defined Networking (SDN) that enables dynamic network programming; and (iii) Service Function Chaining (SFC) that is the process of steering traffic through a set of Virtual Network Functions (VNFs) in an ordered manner to deliver an end-to-end service [1]. Most of the research and standardization efforts that have been made for SFC placement and orchestration focus on the scenario where the chain is deployed on a single administrative domain. However, there are cases where a service requires its functions to be deployed on more than one domain because of resource shortage, strict latency or resiliency requirements, or because of functional considerations. In that case, the SFC is partitioned into sub-chains that are embedded on different domains, then chained together in order to ensure the end-to-end service.

Note that in this work, a domain is defined as an autonomous network infrastructure, with an orchestration entity that independently implements its own management decisions, and networking protocols. Therefore, the multi-domain context can be applied to multiple administrative entities, but also to separate divisions of the same entity. For functional reasons, as well as security concerns, each domain discloses minimal information on its infrastructure [2]. In this scenario, a set of additional constraints for SFC deployment and life-cycle management need to be considered, due to the limited visibility and control on the network infrastructure of each domain, as well as the possible heterogeneity in encapsulation and packet forwarding protocols. In this work, we present a reference architecture that enables the deployment of Service Chains spanning multiple administrative domains. Our contributions are two-fold: First, we introduce a novel architectural framework that performs an end-to-end orchestration of multi-domain SFCs regardless of the internal communication protocols used by each domain. Second, we implement a Proof of Concept of our architecture, and conduct an evaluation of its performance using different encapsulation protocols.

This paper is organized as follows. We first provide a brief background on the existing contributions to the topic in section II. Next, we detail our proposed architectural framework as well as a Proof of Concept and its results in section III before concluding the paper in section IV.

II. Background

The general architecture for Service Function Chaining, including the different functional components of an SFC, their features and interactions, is described in the RFC 7665 [1]. However, this architecture only considers the basic scenario where the SFC is deployed on a single domain. The RFC 8459 [3] attempts to tackle the multi-domain issue by proposing a hierarchical multi-level network architecture for Service Function Chaining that allows the decomposition of a large network into a set of independent sub-domains. A SFC is then decomposed into sub-SFCs, one for each sub-domain, and each sub-SFC is viewed by the top-level as a Service Function along the end-to-end SFC. At the ingress or egress of each sub-domain, the packets are re-classified in order to be forwarded along the corresponding higher level or lower level path. This re-classification is performed by an interfacing function called Internal Boundary Node (IBN) that acts as an SFC-aware Service Function (SF) in the higher-level domain, and as a classifier in the lower-level domain. Nevertheless, the RFC assumes that all the sub-domains are part of the same administrative domain, and does not consider multi-domain issues related to visibility and inter-operability. Indeed, the multi-domain context is much more complex, especially when the domains belong to different administrative entities, which would be reluctant to disclose details on their infrastructure due to security and privacy considerations [2].

On the other hand, multiple contributions proposed orchestration frameworks that extend the ETSI MANO reference

architecture for the multi-domain case. In [4], a reference architecture for multi-domain NFV orchestration is proposed, and an overview of the remaining open issues and challenges is provided. The authors of [5] introduce ETSO (End-To-End SFC Orchestration Framework), a modular ETSI NFV-MANO compliant framework that ensures an end-to-end SFC orchestration. Similarly, Medhat *et al.* [6] propose X-FORCE, an ETSI NFV-MANO compliant SFC orchestration framework that integrates an SFC orchestrator in the reference architecture. However, none of these works considers the specific challenges of SFC orchestration in a multi-domain scenario such as encapsulation, and the inter-domain packet forwarding.

Meanwhile, the ETSI framework for Orchestration and Management (MANO) was extended in its third release [7] in order to support multi-domain orchestration by adding a reference point (Or-Or) for inter-orchestrator (Orchestrator-Orchestrator) communication. The document defines the exchanged information between the orchestrators for service deployment such as Network Service Descriptors (NSDs). However, this information is not sufficient to deploy an end-to-end SFC; indeed, each domain along the chain ought to be aware of the next domain that the packets should be forwarded to at the end of its local sub-chain. Furthermore, since the encapsulation and traffic steering methods are not always the same in every domain, it is necessary to integrate interfacing entities in each domain, that would ensure the end-to-end service delivery regardless of the underlying technologies, by performing inter-domain packet forwarding, encapsulation/decapsulation or tunneling of packets, as well as communication protocol negotiation.

III. Proposed Framework

A. Architecture Overview

In this section, we propose a multi-domain orchestration framework for SFC. Our framework extends existing standardization efforts, namely the third release of the ETSI NFV MANO specification [7] [8], as well as the hierarchical SFC principle proposed by the IETF [3].

Figure 1 illustrates our proposed framework, where each domain possesses its own NFV Management and Orchestration entities according to ETSI's specification [8]. Moreover, the local domain orchestrators are connected to a Multi-Domain Orchestrator (MDO) through an interface that extends the Or-Or reference point. The multi-domain orchestrator is responsible for ensuring the end-to-end deployment of the global SFC, and could also be a local domain orchestrator. In addition to the information that can be shared through the Or-Or reference point, our extended interface supports the transmission of supplementary information that is specific to the multi-domain SFC scenario. Indeed, it enables the MDO to send information that allows the local domains to identify the end-to-end chains, as well as the sub-chains that packets belong to, and perform encapsulation and packet forwarding accordingly.

We leverage on the hierarchical SFC proposal of the IETF [3] by incorporating the Internal Boundary Node (IBN) as the interfacing entity mentioned in the previous section. However, the RFC supposes that the Wide Area Network (WAN) domain (higher level) is SFC-aware, and that inter-domain packet forwarding would be ensured using SFC encapsulation, which is not always the case in the multi-domain scenario. Therefore, we consider in this contribution the case where the WAN domain is not SFC-aware, by adding a mechanism for inter-IBN communication that allows SFC identification.

At the domain's egress, the IBN is responsible for inter-domain packet forwarding, as it strips off the local domain encapsulation of the packets, and sends them to the next domain in the global chain through the border gateways of the local domains. It also adds information that allows the IBN at the receiving end to classify the packets. At the domain's ingress, the IBN acts as a classifier by identifying the sub-chain each packet belongs to, and adding the corresponding local domain encapsulation. In order to perform these operations, each IBN should keep a mapping table of the global service path that each sub-chain belongs to, as well as the next hop after the end of the sub-chain as will be detailed in Section III-C.

Note that thanks to this multi-level architecture, our framework is agnostic to the encapsulation and forwarding protocols used internally by the local domains. Each domain is independent and able to use whichever protocol to ensure the forwarding for its sub-chains as long as the QoS requirements of the Service Level Agreement (SLA) are respected. The IBNs undertake the task of applying the necessary changes to the packet's encapsulation in order to forward them to the other domains. Therefore, the only compatibility that needs to be ensured is the one between consecutive IBNs for inter-domain packet forwarding.

B. Multi-Domain SFC Instantiation Process

We hereby describe the SFC instantiation process over multiple domains. At first, the customer wishing to deploy its SFC sends its NSD to the multi-domain orchestrator; in turn, the multi-domain orchestrator performs an initial placement of the chain by determining the domains where each Service Function will be placed based on the information that has been disclosed by each domain; the request is then partitioned to sub-chains accordingly, and sub-NSDs are created.

The multi-domain orchestrator then creates a mapping between the global service path ID and the position of the sub-chain in that path with the domain where it will be deployed. Taking the example of the Network Service Headers (NSH) encapsulation [9], the path ID would correspond to the Service Path Identifier (SPI), and the position in the path would correspond to the Service Index (SI). The MDO then proceeds by sending to each local domain orchestrator the corresponding sub-NSD, as well as the global path ID that the sub-SFC belongs to, its position, and the identifier of the next domain that the packets should be forwarded to, if applicable. Upon receiving that information, each local orchestrator performs a local placement of its sub-SFC using a local algorithm [10] and deploys it, and instructs its local SDN controller to create the local Service Path of the sub-chain using the sub-VNFFG (VNF Forwarding Graph). The

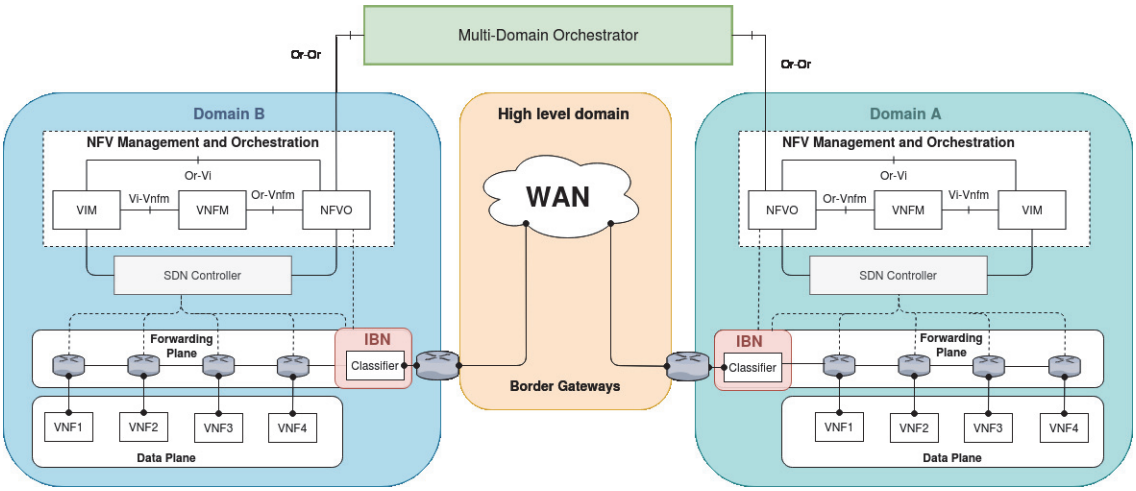


Fig. 1: Architecture for multi-domain SFC orchestration

SDN controller then creates the Service Path, and installs the flow rules on the Service Function Forwarders (SFF) of the domain in order to route the packets according to the Service Path. The controller also installs lower-level classification rules on the IBN in order to classify the packets at the domain's ingress. The controller will then return the local Service Path ID to the local orchestrator that will in turn add an entry to the IBN's mapping table, which is comprised of the global SPI and the position of the sub-SFC, the corresponding local SPI, as well as the next hop if applicable. Afterwards, the local orchestrator confirms to the MDO that the sub-SFC has been deployed. In turn, the MDO confirms to the customer the deployment of the complete SFC after receiving confirmation from each domain.

C. End-to-End Packet Forwarding

In the following, we detail how the end-to-end packet forwarding is ensured. Note that we assume here that each domain is independent in its SFC management (identification, encapsulation, and so on). The multi-domain orchestrator keeps track of where each part of each SFC is deployed, and each SDN controller manages the packet forwarding for the local sub-chains by pushing the forwarding rules into the Service Forwarders; through the SFC instantiation workflow, the IBNs have been made aware of the mapping between the higher level and lower level encapsulations, as well as the next domain to forward the packets to at the end of the sub-chain.

In this example from Figure 1, the SFC packet flow originates from VNF1 in the first domain; it is forwarded by the forwarding plane to the three next VNFs then out to the IBN by relying on the SFC encapsulation, as well as the installed forwarding rules. Once the packets reach the IBN, the latter identifies the global SFC using its mapping table, and determines the next hop for the global SFC, which in that case is the domain B. The IBN then proceeds to strip off the lower-domain encapsulation, and forwards the packets to the border gateway in order to route them to the IBN of domain B. At this point we can identify two possibilities for inter-domain packet forwarding :

a) The WAN domain is SFC aware: In that case the IBN re-encapsulates the original packets using the higher level SFC encapsulation; the border gateways and the WAN domain routers would act as SFFs and forward the packets to the next domain's IBN. The receiving IBN would then act as a SF at the higher level, and as a classifier at the lower level, stripping off the higher level encapsulation then adding the required sub-chain encapsulation upon receiving the packets. In order to perform SFC packet forwarding, the WAN routers would need to be connected to a control plane that manages the higher level chains, such as an SD-WAN controller that would be connected to the MDO.

b) The WAN domain is not SFC aware: A more general and realistic scenario supposes that the SFC encapsulation is not supported by the WAN domain forwarders. In this case, the packets ought to be tunneled between the IBNs of each domain, which supposes that a discovery protocol has been performed in order to identify them, and that at least each pair of consecutive IBNs in the higher level path are able to exchange packets using the same transport protocol. Once an IBN receives the packets, the sub-chain that they belong to is identified through re-classification using the information included by the previous IBN. Then, the corresponding lower level encapsulation is added before forwarding the packets to the first SFF.

D. Proof of Concept

In the following, we depict our Proof of Concept of the proposed framework. Our deployment testbed is built over two physical hosts that represent two different domains. The first domain is hosted on a server with 4 Intel Xeon 2.93GHz CPU Cores and 16GB of memory while the second domain is hosted on a machine with two Intel Core 2.30GHz CPU Cores and 16GB of memory. The MDO is deployed on a third machine that is equipped with two Intel Core 2.40GHz CPU Cores and

8GB of memory. All of the machines host an Ubuntu Server 16.04 x64 OS.

We also leverage on LXC [11] containers that are used in order to host the different Service Functions. They are connected through OVS (Open vSwitch) [12] switches, which ensure packet forwarding using OpenFlow rules. Version 2.11.0 of OVS is used in order to support the NSH encapsulation. The physical hosts are interconnected using VXLAN tunnels. Packet processing is performed by the classifiers, IBNs and VNFs leveraging on a Python-based library *scapy* [13] for packet header manipulation. We also implement our multi-domain orchestrator that performs NSD partitioning, as well as local domain orchestrators that ensure the sub-SFC deployment. We use a simple SDN controller that constructs the classification and forwarding rules for each SFC, and enforce them at the switch level, as well as the classifiers and IBNs at the domain's borders.

In order to support Service Chaining, two different encapsulation types are used: The Network Service Header [9] encapsulation, where the SFC Identifier is encoded into the Service Path Identifier (SPI) field, and the position of the packet inside the chain is encoded into the Service Index (SI) field; and the Segment Routing [14] encapsulation where the classifier encapsulates the forwarding path into a packet as a list of hops; in our implementation, we leverage on Multi Protocol Label Switching (MPLS) headers, where each Service Function ID is encoded into an MPLS label.

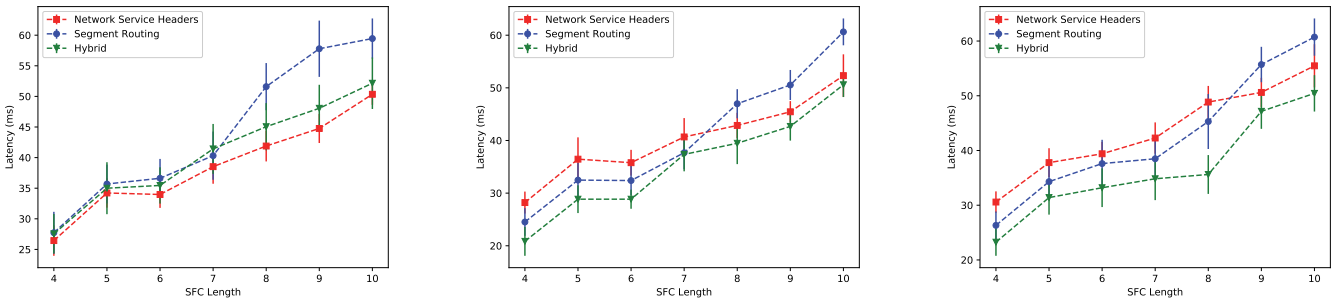
We evaluate our implementation using two metrics. First, we observe the total deployment time of the end-to-end SFCs from the reception of the NSD up to the successful configuration of all of the SFs and forwarding elements (classifier, SFF, IBN). The SFC placement time is disregarded as it depends on the placement algorithm, which has been explored in the literature and is out of the scope of this paper. The second evaluation metric is the end-to-end latency that is added by the packet classification, encapsulation, and forwarding operations. Here, we only consider the encapsulation/decapsulation processing time, and ignore the latency added by the processing of Service Functions, as it is specific to each SF type, and depends on the deployed software.

SFC Length	4	5	6	7	8	9	10
Mean Time (s)	12.40	12.16	16.39	16.44	21.35	21.45	26.82
C.I (s)	0.22	0.26	0.26	0.28	0.34	0.31	0.65

TABLE I: End-to-End SFC Deployment Time

a) End-to-End Deployment time: For this experiment, we generate NSDs for SFCs of lengths that range between 4 to 10 Service Functions. Since our deployment doesn't include the SFC placement process, we assume that half of the SFC is deployed on the first physical machine, and the second half is deployed on the other machine; the classifier and IBN of each domain are also deployed on separate containers, the experiment is repeated 10 times. Table I features the mean deployment times, as well as the 95% Confidence Interval (C.I). The end-to-end deployment time is measured from the reception of the SFC request by the MDO until the reception of the last confirmation message from local orchestrators, which includes the NSD partitioning, as well as the VNF deployment, and flow rules generation and installation. It can be noticed overall that the total deployment time gradually increases with SFC length, but it remains within the range of seconds with a total mean time of *12.40s* and C.I of *0.22s* at the SFC length of 4 VNFs, *16.39s* with a CI of *0.26s* for SFC length of 6, *21.35s* with a C.I of *0.34s* at 8 VNFs and finally *26.69* seconds with a C.I of *0.41s* at SFC length of 10. It is worth noting that the sub-SFC deployment time depends on the machine's hardware configuration, which means that the total deployment time is also affected, since the MDO waits for the confirmation of all of the local orchestrators in order to conclude the end-to-end deployment process.

b) Latency: Once the SFCs have been deployed, we generate 700 packets from each SFC and randomly send them to the first domain's classifier. We then compute the time that it takes for a packet to be received by the last node of its corresponding SFC depending on the SFC length as well as the packet rate. We will also use three different encapsulation approaches: Network Service Headers on both domains, Segment Routing encapsulation based on stacked MPLS labels on both domains, as well as a hybrid approach where each domain uses a different encapsulation. In all scenarios, we assume that the IBN at the end of the first domain strips off the SFC header, and adds an MPLS label that can be used by the second IBN in order to identify the global SFC, as well as the sub-SFCs position, and insert the local domain's corresponding headers. Packets from SFCs of lengths 4 to 10 are generated and sent at rates of 10000, 2500, and 500 packets per second, and the experiment is repeated 10 times. Each SFC can be identified by the classifier using the packet headers and payload. Figures 2a, 2b, and 2c feature the measured latency in milliseconds as well as the 95% Confidence Interval for each packet rate respectively. We can observe that the end-to-end latency ranges between *20ms* and *60ms* depending on SFC length, with slight variations related to the encapsulation type and packet rate. At the packet rate of 10000 packets per second in figure 2a, it can be noticed that the full NSH scenario exhibits the lowest latency values with *26.44ms* for SFC length 4, which will gradually increase to *38.51ms* at SFC length of 7 VNFs and up to *50.34ms* for the SFC length of 10 functions. The hybrid scenario outputs slightly higher latency for the same SFC lengths with mean values of *27.54ms*,*41.42ms* and *52.15ms* respectively. As for the Segment Routing scenario, the latency at SFC length of 4 is *27.73ms*, then *40.31ms* at SFC length of 7 VNFs, then it quickly increases to *59.45ms* at the SFC length of 10. As for the packet rates of 2500 and 500 packets per second in figures 2b and 2c respectively, at first for SFC length of 4 the NSH encapsulation has the highest latency value with mean values of *28.20ms* and *30.57ms* respectively as opposed to *24.51ms* and *26.32ms* respectively for the Segment



(a) Packet Rate 10000pps (b) Packet Rate 2500pps (c) Packet Rate 500pps

Fig. 2: End-to-End SFC Latency Depending on the encapsulation type

Routing encapsulation, and *20.84ms* and *23.24ms* respectively. However, as the SFC length increases, the latency for the last two scenarios increases at a higher rate, thus making the latency for Segment Routing at SFC length 10 reaches the value of *60.63ms* for the packet rate of *2500pps*, and *60.72ms* for the packet rate of *500pps*, while the latency values for the hybrid scenario get closer to the ones obtained with the NSH scenario with latency of *50.58ms* and *52.32ms* respectively for the packet rate of *2500pps*, and *50.44ms* versus *55.47ms* respectively for the packet rate of *500pps*.

This behavior is due to header size, as the total size of the used NSH header (MD-Type 1) is 24 bytes, while the Segment Routing encapsulation relies on stacked MPLS headers of 4 bytes, where each header contains the ID of one SF in the chain, which means that past a sub-SFC length of 6 SFCs, the total packet size using the Segment Routing encapsulation becomes larger than the one using NSH. It can also be observed here that packet rate also affects this difference in latency, since the Segment Routing and hybrid scenario latency values surpass the ones of NSH from the SFC length of 4 for the packet rate of *10000pps*, while for the packet rates of *2500pps* and *500pps* the Segment Routing latency doesn't exceed the NSH one until SFC lengths of 8 and 9 SFCs respectively. The gap between the NSH and hybrid scenario also gradually decreases for both packet rates, but in a slower manner since one of the domains is still using the NSH encapsulation in the hybrid scenario. It can therefore be expected that the latency for the hybrid scenario surpasses the one obtained using the NSH encapsulation for longer SFCs.

Accordingly, the choice of the implemented encapsulation type by each domain should be made based on the length of the sub-SFC that ought to be deployed: a shorter sub-SFC may benefit from the Segment Routing encapsulation as the total encapsulation size would remain minimal, thus ensuring a lower end-to-end latency. However, the NSH encapsulation should be considered for longer SFCs, since the former encapsulation type is not scalable. Additionally, as the header size increases, the total packet size might exceed the Maximum Transfer Unit (MTU), therefore causing fragmentation issues.

IV. Conclusion

In this paper, we introduced a novel framework that enables multi-domain SFC deployment. The architecture is ETSI-MANO compliant, and leverages on the hierarchical SFC principle by using the IBN as an interfacing entity for the local domains. We devised on a technology agnostic end-to-end packet forwarding mechanism that ensures cross-domain consistency, and demonstrate its efficiency with a Proof of Concept implementation. In future works, we aim to extend this solution in order to support post-deployment SFC orchestration operations.

References

[1] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015.
[2] M. Shen, K. Xu, K. Yang, and H. H. Chen, "Towards efficient virtual network embedding across multiple network domains," in *2014 IEEE 22nd Int. Symp. of Quality of Serv. (IWQoS)*, May 2014, pp. 61–70.
[3] D. Dolson, S. Homma, D. Lopez, and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)," RFC 8459, Sep. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8459.txt
[4] K. Katsalis, N. Nikaein, and A. Edmonds, "Multi-domain orchestration for nfv: Challenges and research directions," in *2016 15th International Conference on Ubiquitous Computing and Communications*, Dec 2016, pp. 189–195.
[5] M. Mechtri, C. Ghribi, O. Soualah, and D. Zeghlache, "Nfv orchestration framework addressing sfc challenges," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 16–23, June 2017.
[6] A. M. Medhat, G. A. Carella, M. Pauls, and T. Magedanz, "Extensible framework for elastic orchestration of service function chains in 5g networks," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 327–333.
[7] ETSI GS NFV-IFA 030, "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Multiple Administrative Domain Aspect Interfaces Specification," ETSI, Tech. Rep., April 2019.
[8] ETSI GS NFV-MAN 001, "Network Functions Virtualisation (NFV); Management and Orchestration ," ETSI, Tech. Rep., December 2014.
[9] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," RFC 8300, Jan. 2018.
[10] L. Yala, P. A. Frangoudis, G. Lucarelli, and A. Ksentini, "Cost and availability aware resource allocation and virtual function placement for cdnaas provision," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1334–1348, 2018.
[11] Linux Containers (LXC), https://linuxcontainers.org/.
[12] Open vSwitch, https://www.openvswitch.org/.
[13] Scapy Project, https://scapy.net/.
[14] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," RFC 8402, Jul. 2018.