On Cross-Domain Service Function Chain Orchestration: An Architectural Framework

Nassima Toumi[1,*], Olivier Bernier[1], Djamal-Eddine Meddour[1], Adlen Ksentini[1]

Abstract

Service Function Chaining (SFC) has gained momentum as one of the building blocks of the 5G ecosystem. Indeed, SFC combined with other promising technologies such as Software Defined Networking (SDN) as well as Network Slicing form the basis for enabling the 5G network services and fulfilling their requirements. Many contributions have been made to support and enforce SFC, from chain composition to its deployment and life-cycle management. However, few research works study SFC orchestration, and even fewer consider the multi-domain context, which remains an open issue with many challenges that need to be tackled in order to provide end-to-end services. In this paper, we contribute to filling this gap by introducing a novel architecture for orchestrating and enforcing multi-domain SFCs. The proposed architecture leverages on the ETSI MANO standard, as well as SDN and the hierarchical SFC principles, and is agnostic to the packet forwarding technology used by each administrative domain. We also implement a Proof of Concept (PoC) that employs different SFC encapsulation types to evaluate the architecture, where the obtained results prove our architecture's efficiency, we're also able to draw conclusions related to the encapsulation methods to implement. Finally, we discuss open research issues.

Keywords: SFC, Multi-Domain, Management and Orchestration

1. Introduction

The upcoming generation of networks (5G) is set to accommodate new heterogeneous use cases with specific requirements, such as enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC); in order to do so, new technologies and paradigms have emerged, namely Network Function virtualization (NFV) that allows a better flexibility in management, Software Defined

*N. Toumi (nassima.toumi@orange.com), O. Bernier (olivier.bernier@orange.com), and D.E. Meddour (djamal.meddour@orange.com), are with Orange, 22300, Lannion, France.

**N. Toumi and A. Ksentini (adlen.ksentini@eurecom.fr) are with the Communication Systems Department, EURECOM, 06410 Sophia-Antipolis, France.

*Corresponding author

Networking (SDN) which enables dynamic network programming, and Service Function Chaining (SFC) which is the process of steering traffic through a set of functions in an ordered manner to deliver an end-to-end service. Many research and standardization contributions have been made to enable the SFC paradigm, but most of these efforts have focused on the scenario where the SFC is deployed on a single administrative domain. However, there are cases where a service requires that its functions be deployed on more than one domain, due to the lack of resources, strict latency or resiliency requirements, or functional considerations. Among the scenarios of multi-domain SFCs, we can cite the Edge deployment use cases such as Industrial IoT [1], or autonomous vehicles [2], where sensor data is first pre-processed by functions deployed at the Edge of the network, before being sent to the Core network for further analysis. The service operator might not dispose of sufficient Edge servers to cover the whole sensing area. A cost-efficient solution would then be to deploy the Edge part of the SFC on the Edge servers of an IaaS provider, and forward the traffic back to the core cloud of the service operator. A more generic use case for multi-domain SFCs leverages on the Function-as-a-Service cloud service model, where the service tenant doesn't own any infrastructure, but composes its custom SFC using Service Functions that are provided by different FaaS operators, and chained together to create the end-to-end multi-domain SFC [3].

. The multi-domain scenario introduces a set of new challenges. Indeed, due to security considerations, administrative domains are reluctant to divulge details on their infrastructure, and do not grant full control over the SFCs that may be deployed on their domain; thus offering limited visibility and control to the SFC owner. This requires adaptations of the procedures of automated management and configuration of the physical and virtual resources, and of the life-cycle management of the Service Chain, also referred to as SFC orchestration. Furthermore, since each domain operates independently, interoperability issues may arise due to possibly different encapsulation and packet forwarding protocols, calling for the implementation of additional measures in order to guarantee compatibility and perform packet forwarding along the SFC from end to end. In this work, we tackle these issues by presenting a novel reference architecture for SFC orchestration that enables the deployment of Service Function Chains spanning multiple administrative domains, agnostically to the SFC encapsulation methods that are implemented by each domain. Note that in the following, the terms multi-domain and cross-domain can be used interchangeably.

. This paper's contributions are three-fold : first, we propose an orchestration architecture that leverages on existing standardization efforts in order to ensure an end-to-end orchestration of multi-domain SFCs regardless of the internal communication protocols used by each domain, with a multi-domain SFC deployment protocol that performs the necessary configurations to enable the end-to-end packet forwarding. Second, we implement a PoC of our architecture, and perform an extensive evaluation based on various Key Performance

Indicators (KPI) using different encapsulation protocols. Third, we provide an analysis of open issues for future research works.

This article is organized as follows. We first provide a brief background on the existing contributions to the topic in section 2. Next we describe our proposed architecture in section 3, and the PoC implementation and its evaluation results are detailed in section 4. We then proceed by discussing open issues in section 5 before concluding the paper in section 6.

2. Background and Motivation

The general architecture for Service Function Chaining is described in the RFC 7665 [4]. It comprises Service Functions (SF) that perform specific treatment of ingress traffic, Service Function Forwarders (SFF) that are responsible for steering the traffic between the SFs according to a pre-defined Service Function Path (SFP) which is equivalent to ETSI's VNF Forwarding Graph (VNFFG)[5], and classifiers at the ingress of each SFC domain that perform classification in order to determine the SFC that a packet belongs to, and encapsulate it accordingly. SFC proxies can also be used in case some Service Functions don't support SFC encapsulation. However, this architecture only considers the basic scenario where the SFC is deployed on a single domain. 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, and operate their domains independently.

Multiple contributions have been made in multi-domain service orchestration, multi-domain SFC, and SFC packet forwarding. They can be classified as follows:

2.1. Multi-Domain SFC Placement

A number of contributions [6, 7, 8, 9, 10, 11] studied SFC placement with limited visibility, where two main architectural approaches are employed. With the first approach, a distributed algorithm is executed on all domains, and messages are exchanged in order to determine the best option without disclosing information to external parties. The second approach allows a logically centralized coordinator to collect information disclosed by the domains, and reconstitutes an abstract global view of the network in order to perform placement. However, these works only study the multi-domain SFC placement problem, which is only a part of the SFC orchestration process, and is not sufficient for ensuring the effective end-to-end deployment, as it requires additional steps such as the configuration of the different SFC components, and the required information exchange. Other contributions [12, 13] have also explored multi-domain Virtual Network Embedding (VNE), which has similarities with SFC placement, but is different. Indeed, the VNE problem concerns the embedding of undirected graphs, and doesn't consider the order constraints of SFCs, which can affect the end-to-end latency [14, 15]. Furthermore, similar to the previously mentioned works, these contributions only consider the placement phase.

2.2. Multi-Domain Service Orchestration

Multiple contributions proposed orchestration solutions that extend the ETSI MANO [5] reference architecture. The ETSI framework for Orchestration and Management (MANO) has been extended in its third release [16] in order to support multi-domain orchestration by adding a reference point (Or-Or) that allows orchestrators to communicate. The document defines the exchanged information format between the orchestrators for service deployment such as Network Service Descriptors (NSD). Nonetheless, this information is not sufficient to deploy an end-to-end SFC; indeed, each domain along the chain ought to be aware of the global SFC that its sub-SFC belongs to, in order to properly identify the packets at its ingress. The local domains should also be aware of the next domain that the packets should be forwarded to, so that the end-to-end packet steering is ensured. In [17], a reference architecture for multi-domain NFV orchestration is proposed, and an overview of the remaining open issues and challenges is provided. Similarly, an orchestration architecture for multi-domain Network Slice deployment and resource management is depicted in [18]. However, these works don't consider SFCs and the constraints related to packet processing order.

2.3. SFC Orchestration

The authors of [19] introduce ETSO, a modular ETSI NFV-MANO compliant framework that ensures an end-to-end SFC orchestration. Likewise, Medhat *et al.* [20] detail X-FORCE, an ETSI NFV-MANO compliant SFC orchestration framework that integrates an SFC orchestrator in the reference architecture, and Ding *et al.* [21] present OpenSCaaS, a platform that leverages on SDN and NFV to provide SFC as service. Nevertheless, these works ignore the issues related to the multi-domain aspect of SFC deployment, such as the end-to-end packet forwarding and identification at each domain, or the possible heterogeneity in packet encapsulation and traffic steering methods. In [22], the authors present Cloud4NFV, a platform for SFC orchestration across data-centers; the solution adopts a non-tagging classification approach, in which classification is performed at each hop of the SFC. However, this approach generates considerable latency as each packet needs to be classified at each hop. Furthermore, it considers that the distributed data-centers belong to the same domain. An IETF draft [23] proposes two methods for inter-data-center SFC deployments: One with multiple SFC domains where each domain's management is constrained to itself, meaning that each domain is only aware of the sub-SFCs that are deployed within its own infrastructure, as well as the next hop, and is unaware of the higher-level SFC details and configuration. And the second method, with a single SFC domain, where a single logical entity manages the SFC across domains. This approach supposes that the local domains delegate enough control and visibility to the centralized coordinator in order to operate the multi-domain SFC from end to end, which might not always be possible.

2.4. Multi-Domain SFC Packet Forwarding

As for SFC packet forwarding, multiple traffic steering methods have been proposed: header-based methods, such as Network Service Headers (NSH) [24, 25], tag-based methods [26] [27], and programmable switch-based methods [28]. A comprehensive survey on traffic steering methods for SFC can be found in [29]. However, these solutions enable SFC packet forwarding inside a single domain only. The RFC 8459 [30] attempts to tackle the multi-domain issue by proposing a hierarchical multi-level network architecture for SFC that allows the decomposition of a large network into a set of independent sub-domains. Each sub-domain 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. Another draft [31] proposes a horizontal approach for multi-domain NSH. SFCs that span multiple domains are divided into a set of single-domain segments, and the identifier of the next domain's classifier is encapsulated in the metadata part of the NSH encapsulation. However this architecture can only be applied to NSH-aware domains. Indeed, 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.

At the time of this writing and to the best of our knowledge, this paper is the first contribution that addresses all of the aforementioned issues related to multi-domain SFC. Namely, this work tackles the end-to-end SFC deployment and configuration, inter-domain packet forwarding, and the heterogeneity in encapsulation methods.

3. Proposed Architecture

3.1. Overview

In this section, we describe our proposed multi-domain orchestration architecture for Service Function Chaining. It extends existing standardization efforts, namely the third release of the ETSI NFV MANO specification[5] [16], as well as the hierarchical SFC principle proposed by the IETF[30]. Note that in this contribution, the multi-domain context applies to cases where SFCs are deployed across domains that belong to different administrative entities, but also to cases where the domains belong to the same administrative entity, but are operated independently for functional reasons, or for security concerns. This can be the case for branches of the same company for example. We assume

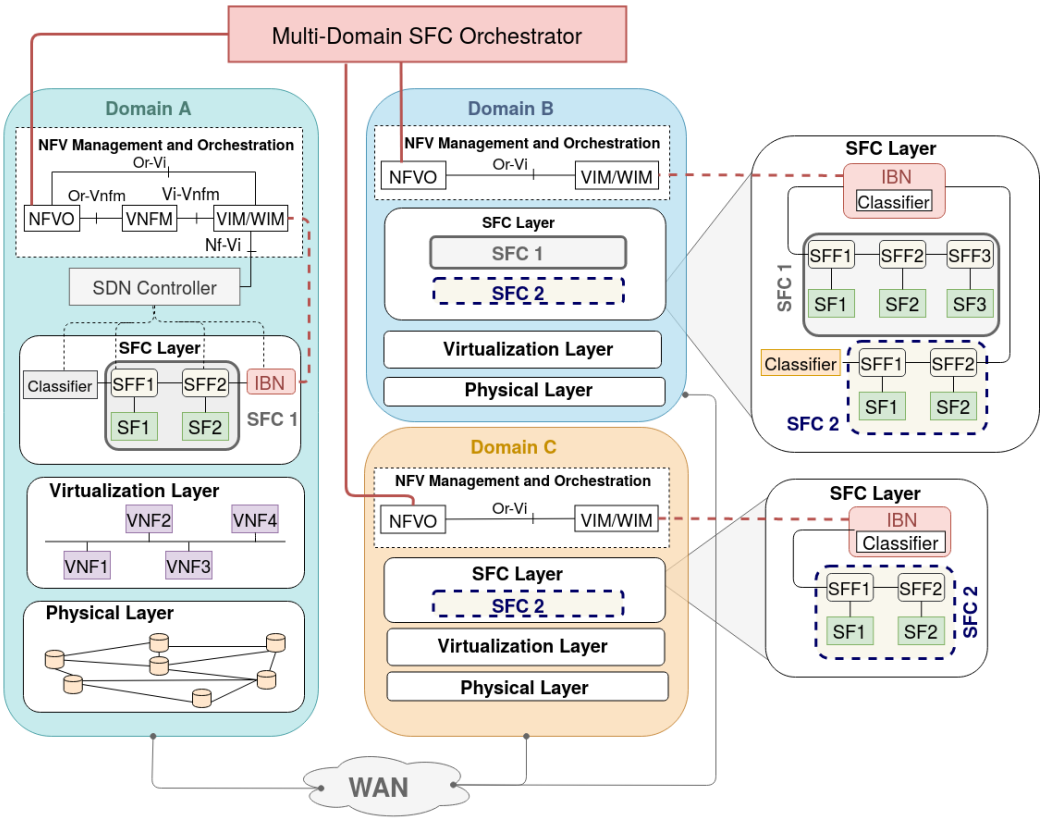


Figure 1: Architecture for Multi-Domain SFC Orchestration

that prior to SFC deployment, an enrollment procedure is performed by the domains in order to participate in the multi-domain SFC deployment, under the supervision of the multi-domain orchestrator. We suppose that this procedure is performed by the administrative entities, where mutual agreements are made in order to set the amount of resources made available by the local domains, as well as the cost per unit. SLAs and security requirements may also be negotiated between the operators. Further, this procedure is also meant to perform authentication, and establish mutual trust between the different entities, as the local domains are delegated partial control over the SFCs. At the end of the enrollment procedure, secure communication channels are established between the orchestrators.

Figure 1 illustrates our proposed architecture, where two SFCs are deployed on three independent domains, the first SFC spans domains A and B while the second SFC is deployed on domains B and C. At the physical layer level, each domain disposes of a set of computing and forwarding elements that are

connected through the physical network. At the virtualization layer, Virtual Network Functions (VNFs) are deployed on the computing nodes and are connected through Virtual Links, these VNFs will act at the SFC layer as Service Functions, IBNs, and classifiers, and the SFC packet steering is ensured by the Service Function Forwarders (SFFs). Traffic steering relies on SDN, where the SDN controller of each domain is responsible for the configuration of each network element in order to perform the desired forwarding of packets. For the sake of brevity, these layers are only detailed for the first domain. We highlight in red the components and interfaces that are added or extended by our proposal. Namely, the multi-domain orchestrator and its interfaces with the local domain orchestrators, as well as the IBN and its configuration interface.

Local Domain Orchestration:. At the orchestration level, each domain possesses its own NFV Management and Orchestration entities as defined by ETSI's specification, and is independent in its SFC management (identification, encapsulation, and so on). Therefore, we consider that each domain is only aware of its local sub-SFCs and the next hop, and ignores details on the global SFC and its other sub-chains. Note that although the Virtual Infrastructure Manager (VIM) and WAN Infrastructure Manager (WIM) are separate entities, they are connected to the other MANO blocks through similar reference points. Therefore, for the sake of simplicity, we represent them as one block. Moreover, the local domain orchestrators are connected to a Multi-Domain Orchestrator (MDO) through an interface that extends the Or-Or reference point, as it supports the exchange of additional information that is required for multi-domain SFC deployment, and that isn't supported by that reference point (see Section 3.3).

Multi-Domain Orchestrator. The MDO is responsible for ensuring the end-to-end deployment of the global SFC, and could also be a local domain orchestrator. During the SFC deployment, the MDO first performs an initial placement of its VNFs, and partitions the NSD accordingly. Then, the sub-NSDs are sent to each local domain orchestrator, along with additional information in order to configure the local IBNs, as will be further discussed in Section 3.2. That information allows the IBNs to perform the inter-domain packet forwarding, by identifying the SFC of the incoming packets, as well a the next hop at the end of the SFC.

Internal Boundary Node. We leverage the hierarchical SFC proposal of the IETF [30] by incorporating the IBN as the interfacing entity mentioned above. However, the RFC supposes that the WAN domain (higher level) is SFC-aware, which is not always the case in a multi-domain scenario. Therefore, we also study in this contribution the case where the WAN domain is not SFC-aware, which means that our IBN implements a different method for the inter-domain packet steering: At the domain's egress, the IBN strips off the local domain encapsulation of the packets, and sends them to the next domain's IBN in the global chain through the border gateways. And at a domain's ingress, the IBN

acts as a classifier by identifying the global chain and sub-chain that each packet belongs to, and adding the corresponding local domain encapsulation. In order to perform these operations, the 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 illustrated in Section 3.3. This mapping table is constructed using information obtained from the MDO, and the local NFVO through WIM and VIM, respectively.

It is important to highlight that the classification that is performed by the ingress IBNs is different than the one performed by the classifier at the beginning of the chain. Indeed, the classifier receives the original packets, and determines the SFC (first sub-SFC in our case) that they belong to using information that is provided by the client, then encapsulates them accordingly. On the other hand, the ingress IBN performs classification using the encapsulation that is inserted by the egress IBN of the previous domain, and that is either determined by the MDO, or from an inter-IBN negotiation protocol. Furthermore, the SFC classifiers are configured by the local NFVO through the VIM, while the classification components of the IBNs are configured by the WIM, which is responsible for the WAN (inter-domain) configuration. Therefore, we consider them as two separate logical entities as illustrated in domain B.

. Thanks to this multi-level setting, our architecture 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 would undertake the task of applying the necessary changes to the packet's encapsulation in order to forward them to the outer domains. Therefore, the only compatibility that needs to be ensured is the one between consecutive IBNs for inter-domain packet forwarding.

3.2. Multi-Domain SFC Instantiation Process

We hereby describe the SFC Instantiation process over multiple domains. Figure 2 illustrates the workflow that takes place when instantiating a multi-domain SFC. At first, the customer, who wishes to deploy its SFC, sends its Network Service Descriptor (NSD) to the multi-domain orchestrator; in turn, the latter will perform an initial placement of the chain, that will determine the domains where each Service Function will be placed based on the information that has been disclosed by each domain as will be discussed in Section 3.2.1. Once the initial placement has been determined, the request is partitioned to sub-chains depending on the domain where each VNF of the SFC has been assigned as will be detailed in Section 3.2.2. Once the sub-NSDs are created, and the global service path is determined, the multi-domain orchestrator 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 [25], 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 multi-domain orchestrator 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 in the SFC, 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 and deploys it, then instructs its local SDN controller to create the local Service Path of the sub-chain using the sub-VNFFG. The SDN controller then creates the Service Path, and installs the flow rules on the Service Forwarders of the domain in order to route the packets according to the Service Path. 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 that is comprised of the global Service Path ID and the position of the sub-SFC, the corresponding local Service Path ID, and the next domain to send the packets to, if applicable. These mappings will allow the IBN to identify the packets at its ingress on one hand, and on the other hand, to forward the packets to the next domain at the end of the sub-chain. Afterwards, the local orchestrator confirms to the multi-domain orchestrator that the sub-SFC has been deployed. In turn, the multi-domain orchestrator will confirm the deployment of the complete SFC after receiving confirmation from each domain.

3.2.1. Request Placement

Request placement is performed by both the MDO and the local domain orchestrators. During the first placement, the MDO determines where each VNF of the SFC should be placed, using information from the NSD, such as the resource and functional requirements of VNFs, or the deployment constraints. Regarding the local placement of the sub-SFCs by each domain orchestrator, the sub-NSD information is also taken into account. Note that this placement is performed by dedicated algorithms and methods, which have been largely studied in the literature, and are out of the scope of this paper. For example, the centralized multi-domain SFC placement solutions that have been proposed in [6, 7, 8] can be used to perform SFC placement on both levels (MDO and local orchestrators). VNF, VL, and CP counters are reinitialized upon the creation of each new sub-NSD; adding a VNF, VL, and CP to a sub-NSD implies copying their descriptors from the original NSD, and adding their IDs to the VNFFG descriptor accordingly.

3.2.2. Request Partitioning

The NSD partitioning process is detailed in Figure 3 and is performed as follows: The consecutive VNFs that have been placed on the same domain are grouped in order to create sub-NSDs for each domain, and external connection points are added at their extremities in order to link those sub-SFCs to the rest of the global SFC, and forward the traffic out of the local domain. Based on these sub-SFCs and connection points, new sub-NSDs are created as illustrated in figures 4 and 5.

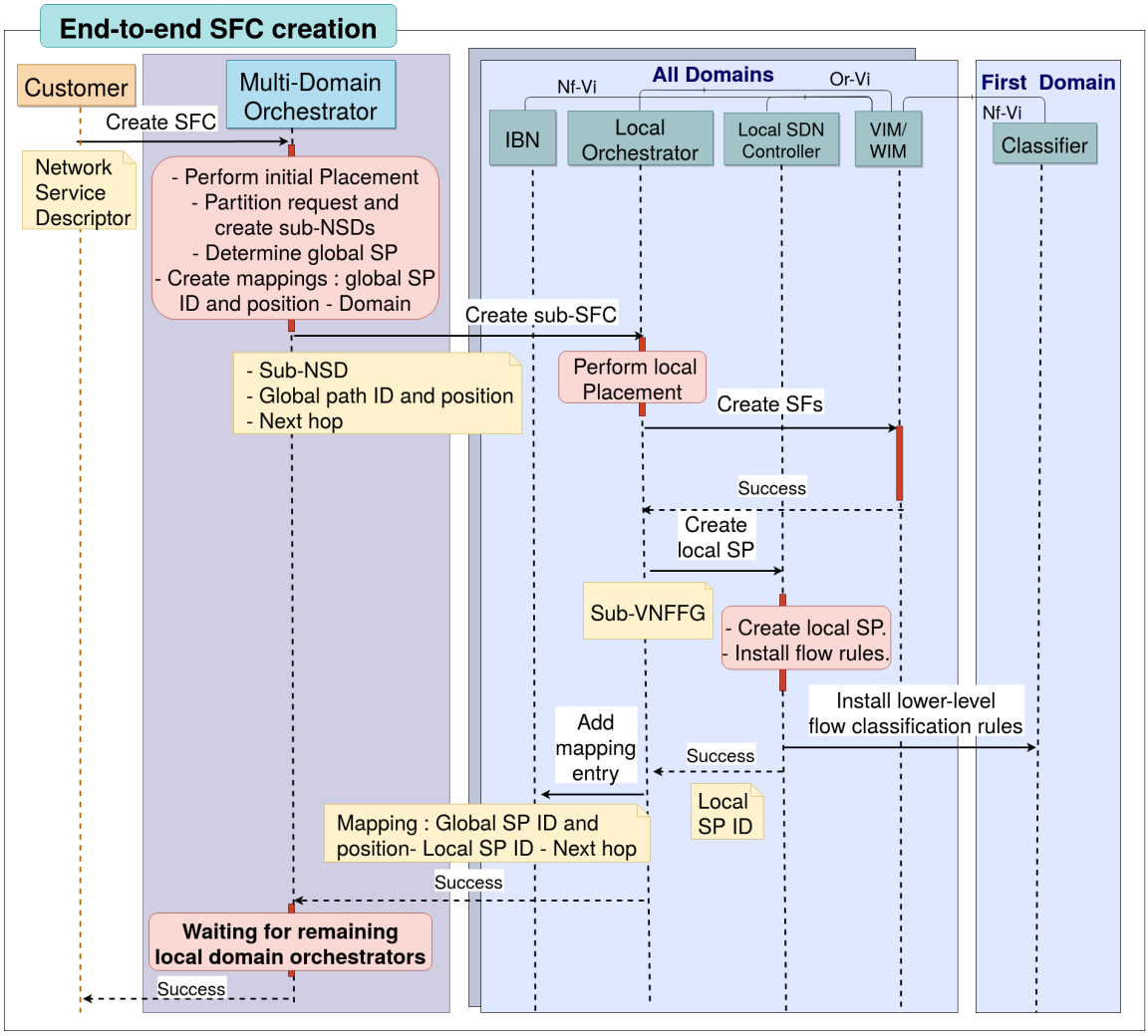


Figure 2: Multi-domain SFC instantiation workflow

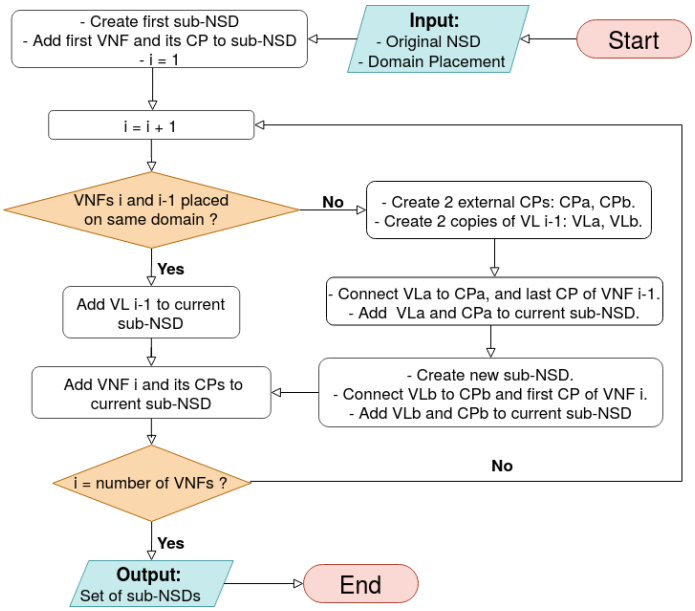


Figure 3: NSD Partitioning workflow

. In this example, we provide two examples for NSD partitioning, depending on the SFC placement. The NSD is modeled using the standardized Topology and Orchestration Specification for Cloud Applications (TOSCA) data model [32], which is also simplified in order to only keep the information that is relevant to our case, namely the VNF Forwarding Graph (VNFFG) and the Forwarding Path (FP). As illustrated in Figure 4, the original SFC comprises 3 VNFs which are connected by Virtual Links VL1 and VL2 through their respective Connection Points: CP11 for VNF1, CP21 and CP22 for VNF2, and CP31 for VNF3, the Forwarding Path is composed of an ordered list of the Connection Points the packets are forwarded to. After the initial placement has been performed by the orchestrator, we consider two placement possibilities : in the first scenario, we suppose that VNF1 and VNF2 are mapped to one domain, and VNF3 is mapped to a second one; the second scenario supposes that each VNF is mapped to a different domain. According to that placement, the original NSD is then partitioned as shown in Figure 5: each sub-SFC except the last one ends in an external Connection Point that will ensure that the packets are routed out to the domain's IBN, and every sub-SFC except the first one will start with an external Connection Point that will forward the packets from the domain's IBN. Note that each domain's respective IBNs are not included in the NSD as they are not deployed with the SFC, indeed, they are deployed by the domain in order to serve all of the SFCs. Furthermore, this NSD partitioning essentially depends on the output of the SFC placement process. Indeed, for other SFC placement scenarios, the NSD would have also been partitioned differently.

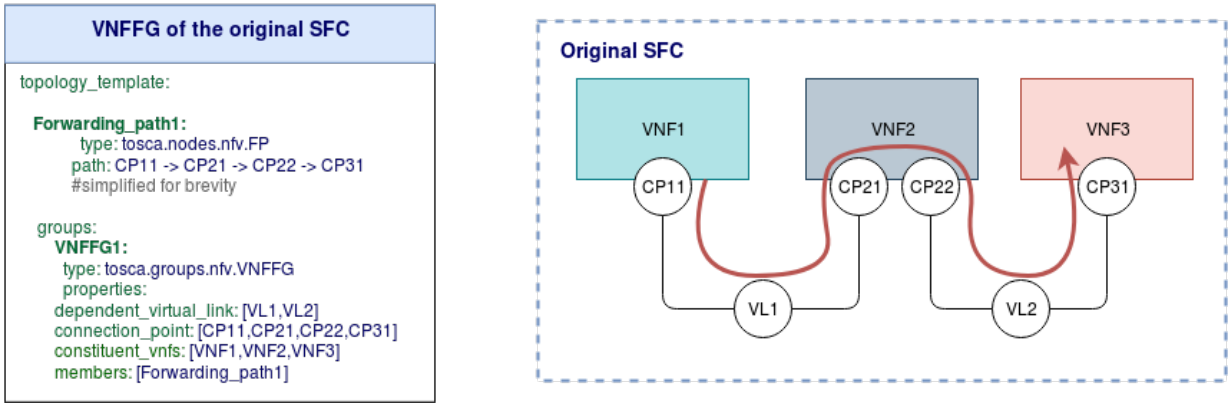


Figure 4: Original SFC and its VNFFG before partitioning

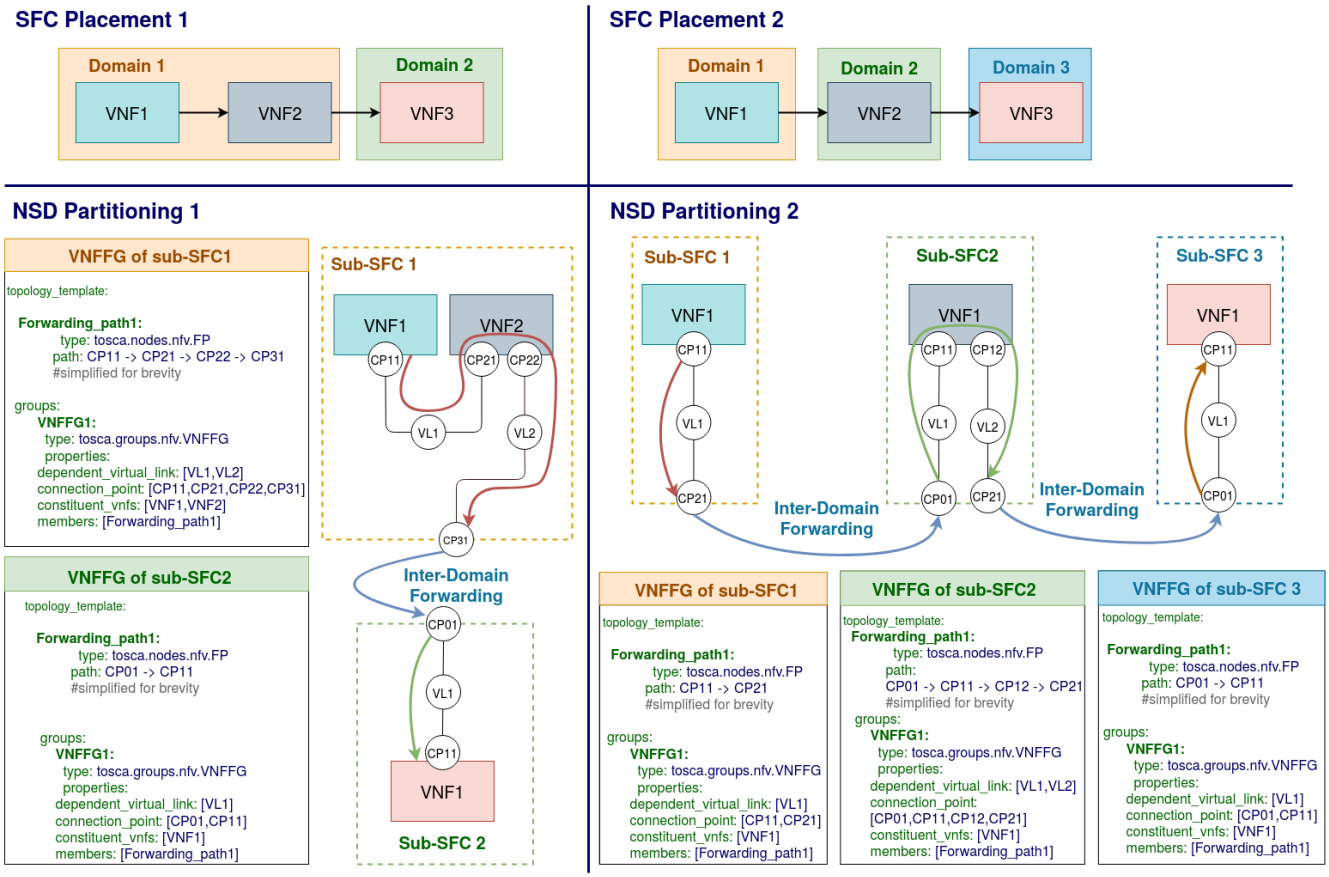


Figure 5: Sub-SFCs and their VNFFGs after the partitioning depending on the SFC Placement

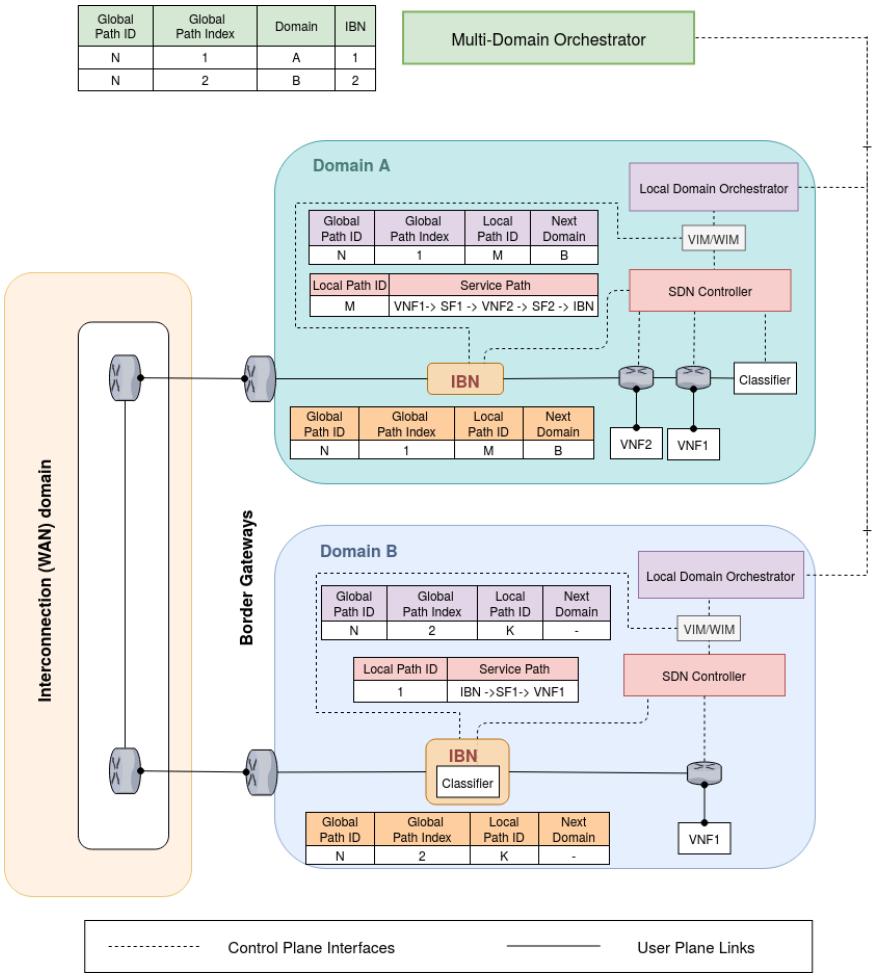


Figure 6: End-To-End SFC Packet Forwarding Mechanism

3.3. End-to-End Packet Forwarding

In the following, we detail how the end-to-end packet forwarding is ensured. Figure 6 shows the color-coded mapping tables of the different components in our architecture after a Service Chain's deployment. 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 SFFs; 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, the SFC packet flow originates from VNF1 in the first domain, it is forwarded by the SFFs to the second VNF then out to the IBN by relying on the NSH encapsulation, as well as the in-

stalled forwarding rules. Once the packets reach the IBN, the latter identifies the global service path and index, and using its mapping table, 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 forward 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 :

The WAN domain is SFC aware. In this case, the IBN re-encapsulates the packets using the higher level SFC encapsulation, and the border gateways as well as the WAN domain routers would act as SFFs, and would 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 inter-domain forwarders 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 multi-domain orchestrator.

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 that 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 the IBNs of each domain, and that at least each pair of consecutive IBNs in the higher level path are able to exchange packets using the same protocol. Once an IBN receives the packets, the sub-chain that they belong to is identified through re-classification, and the corresponding lower level encapsulation is added before forwarding to the first SFF. However, since the packets are processed by different VNFs along the path, the packet state at the egress of a domain's sub-chain might differ depending on the higher-level placement performed by the MDO. Indeed, VNF processing of packets alters their content, which means that the payload and/or header fields of packets at the egress of a VNF *N* are different than those of packets at the egress of the VNF *N+1*, thus making it difficult to keep track of the changes. Therefore, the classification at an IBN's ingress can be performed in two manners :

- Determine the classification rules for each IBN depending on the higher-level placement, which supposes that the MDO is aware of the state of the packets at the egress of each Service Function of the chain.
- Include in the inter-domain packet encapsulation information that allows the IBN to identify the global SFC of the packets and their position.

In the PoC of this work, we assume that the WAN domain is not SFC aware, and that the sub-SFC of each packet at a domain's ingress is determined by including the higher-level SFC ID in the inter-IBN encapsulation, which can be translated to the sub-chain encapsulation using the IBN's mapping table.

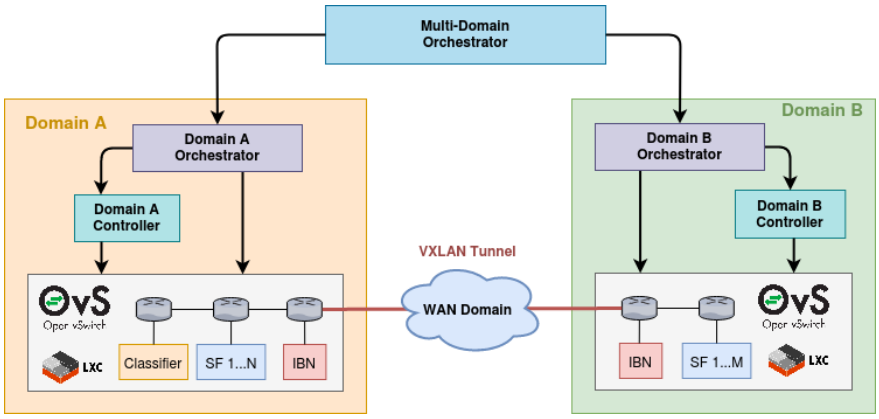


Figure 7: Proof of Concept Testbed Implementation

4. Implementation

Next, to validate our proposal, we implement a Proof of Concept of the proposed architecture to deploy multi-domain SFCs, and run multiple experiments in order to evaluate the performance of the components that have been deployed.

4.1. Testbed Setup

In the following, we depict our Proof of Concept of the proposed architecture. As illustrated in Figure 7, our deployment testbed is built over two physical hosts that represent two different domains. Note that in the following, the terms domain and server can be used interchangeably. We also leverage on LXC [33] containers that are used in order to host the different Service Functions connected through OVS (Open vSwitch) [34] switches that ensure packet forwarding using OpenFlow rules. Version 2.11.0 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 using the Python-based library *scapy* [35], and the traffic is generated using the *hping3* packet assembler [36]. We implemented, using *Python* scripts, our multi-domain orchestrator that performs NSD partitioning, as well as local domain orchestrators that ensure the sub-SFC deployment. In addition, we used a basic SDN controller that will construct 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.

The hardware and software setup is outlined in Table 1.

In order to support Service Chaining, two different encapsulation methods are used, a header-based encapsulation with the NSH header, and a tag-based encapsulation with MPLS Segment Routing:

- The Network Service Header [25] encapsulation is illustrated in Figure 8, the SFC Identifier is encoded into the 3-byte Service Path Identifier

Multi Domain Orchestrator	
CPU	Intel Core (2 Cores), 2.40GHz
RAM	8GB
OS	Ubuntu 16.04, 64 bits
Domain 1	
CPU	Intel (32 Cores), 2.60GHz
RAM	128GB
OS	Ubuntu 16.04, 64 bits
Domain 2	
CPU	Intel Xeon (4 Cores), 2.93GHz
RAM	16GB
OS	Ubuntu 16.04, 64 bits

Table 1: Testbed Hardware and Software Configuration

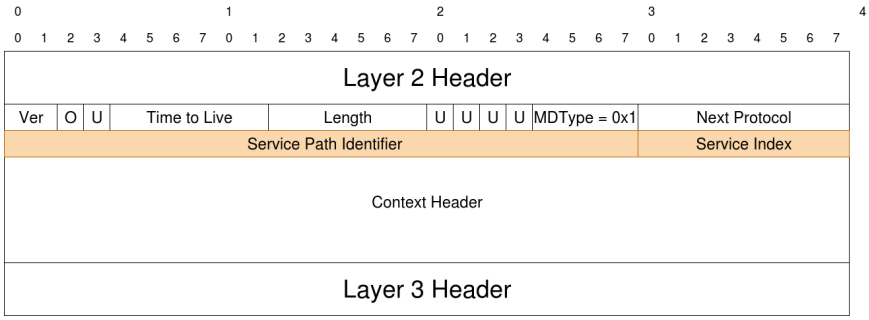


Figure 8: Network Service Header Encapsulation

(SPI) field, and the position of the packet inside the chain is encoded into the Service Index (SI) field; this tuple allows the SFFs to determine the next hop for the packets. This header is added upon the classification of packets, and each time the packet passes through a Service Function along the chain, the SI is decremented. In this implementation, we use the NSH MD-Type 1, which is the fixed-length metadata type, therefore, the Network Service Header's size is fixed to 24 bytes.

- The Segment Routing [37] encapsulation is depicted in Figure 9, where the classifier encapsulates the forwarding path into a packet as a list of hops, therefore reducing the amount of state information that needs to be stored on the forwarding devices. In our implementation, we leverage on Multi Protocol Label Switching (MPLS) headers, where each Service Function ID is encoded into a 20-bit MPLS label. The SFFs determine the next hop by reading the label field of the outer MPLS header, and each time a packet passes by a Service Function, the outer MPLS header is stripped off, so that the packet can be routed to the next SF in the SFC.

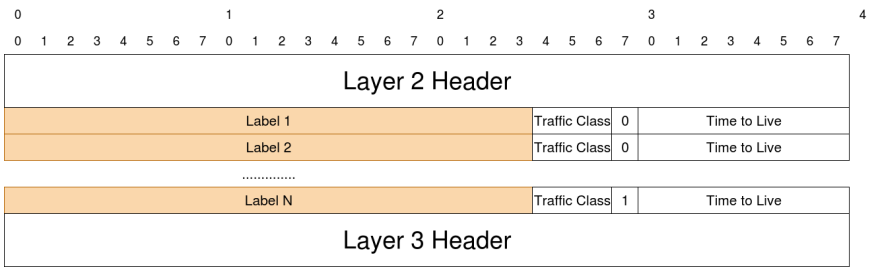


Figure 9: MPLS-Based Segment Routing Encapsulation

The MPLS header is 4-bytes in size, which means that the encapsulation size at any position in the SFC is $4 * n$ bytes, with n being the number of remaining SFs that the packet has to be forwarded to. Therefore, the encapsulation size decreases as packets get further into the Service Chain.

We evaluate our implementation using four key metrics. First, we observe the overall 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 is dependent on the placement algorithm, which has been explored in the literature and is out of the scope of this paper. The second evaluation metric will be the end-to-end latency for each SFC length and each encapsulation scenario; additionally, we measure the delay added by the encapsulation and forwarding of the packets. We will therefore ignore the time consumed by the specific processing of each Service Function and only consider the encapsulation/decapsulation processing time. Finally, we observe the CPU load that is generated from the processing of packets by the SFC components.

4.2. End-to-End Deployment time

For this experiment, we generate NSDs for SFCs of lengths that range between 4 to 30 Service Functions. Since our deployment doesn't include the SFC placement process (out of the paper's scope), we set pre-defined placement combinations for each SFC. Each SFC of 4 to 20 VNFs is split in equal halves between the domains, and for the other SFCs, due to hardware limitations, the first 10 VNFs are deployed on the first domain, and the remaining VNFs on the second one. The classifier and IBN of each domain are also deployed on separate containers. The experiment is performed for 20 iterations. Figure 10 features the average deployment times as well as the 95% Confidence Interval (C.I). Furthermore, the green bars represent the mean time for the orchestration operations, which encompasses the NSD partitioning by the MDO, the sub-NSD file transfer to the local orchestrators, as well as the Service Function, Classifier and IBN configuration by the local domains; while the orange bars represent the

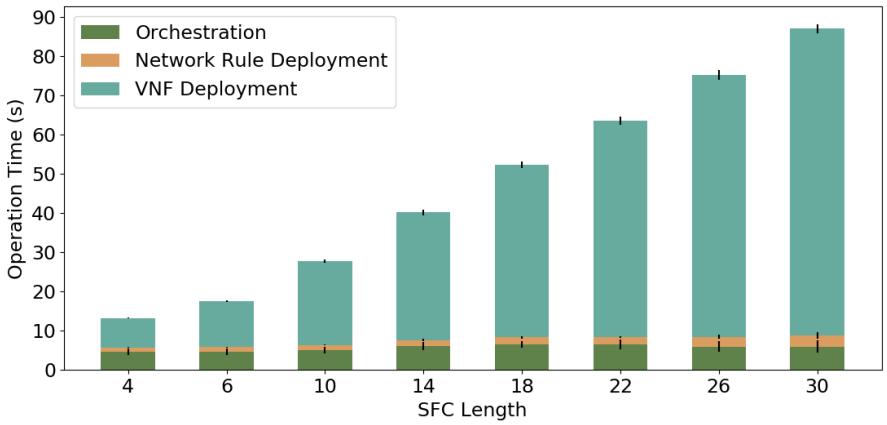


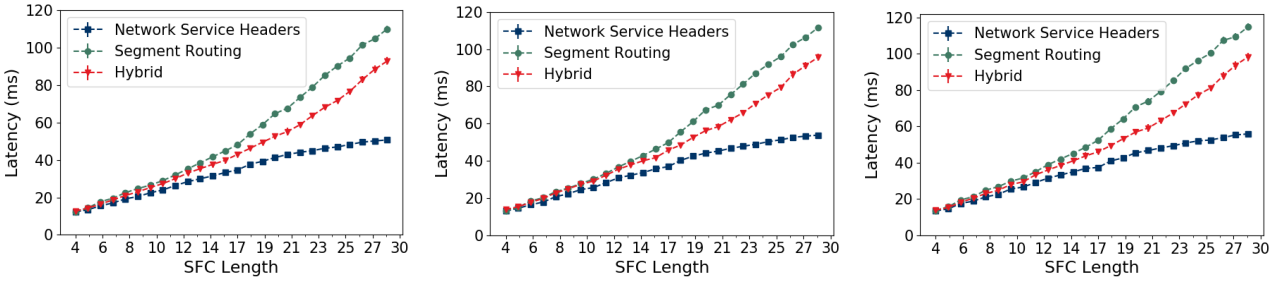
Figure 10: End-to-End SFC Deployment Time

OVS configuration time, and the blue bars illustrate the mean VNF deployment time at the VIM level. The total end-to-end deployment time is the sum of all three bars, and can be measured as the time elapsed from the reception of the SFC request by the MDO until the reception of the last confirmation message from the local orchestrators, as illustrated in Figure 2.

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 average time of *13.29s* at the SFC length of 4 VNFs, and up to *86.93* seconds at SFC length of 30. By breaking down these total times, it can be observed that the VNF deployment process makes up for the largest portion of the SFC deployment time. Indeed, it represents *56.7%* of the total time, with *7.5s* for SFC length 4 and up to *78.16s* representing *89.91%* of the total deployment time at SFC length of 30. In contrast, the OVS rule deployment process only makes up for *3-8%* of the total time, ranging from *1.16s* at SFC length 4, up to *2.79s* at SFC length of 30 VNFs. The remaining time is consumed by the orchestration operations with values that remain stable, ranging between *4.5s*, representing *33.86%* of the total time, and *6.5s* making up for *7.4%* of the total deployment time of the SFC. It is worth noting that since the sub-SFC deployment is performed by each domain in parallel, the MDO waits for the confirmation of all of the local orchestrators before concluding the end-to-end deployment process, which means that the total deployment time is affected by the hardware configuration of the least-performing domain.

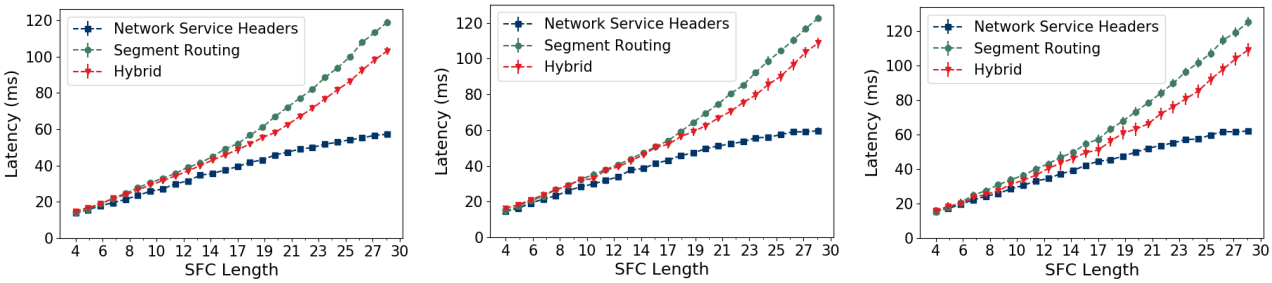
4.3. End-to-End Latency

For this metric, we deploy different numbers of SFCs with lengths of 4 to 30 VNFs, where multiple combinations of SFs are used in order to generate the



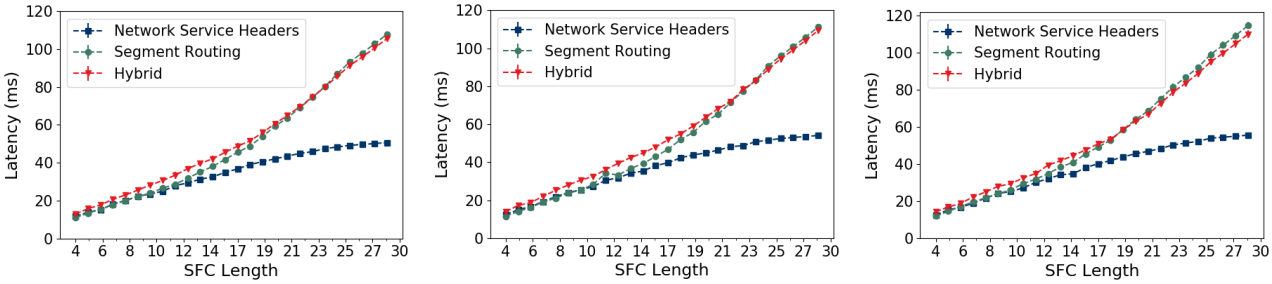
(a) Traffic Rate = 2MB/s (b) Traffic Rate = 10MB/s (c) Traffic Rate = 50MB/s

Figure 11: End-to-End Latency with 50 SFCs



(a) Traffic Rate = 2MB/s (b) Traffic Rate = 10MB/s (c) Traffic Rate = 50MB/s

Figure 12: End-to-End Latency with 150 SFCs



(a) Traffic Rate = 2MB/s (b) Traffic Rate = 10MB/s (c) Traffic Rate = 50MB/s

Figure 13: End-to-End Latency with 450 SFCs

SFPs, and VNF sharing between SFCs is enabled, meaning that a given VNF can serve as a Service Function for multiple SFCs. For this experiment, we evaluate the end-to-end latency for 50, 150, and 450 SFCs.
Once the SFCs have been deployed, we generate packets from each SFC and send them to the first domain's classifier, 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 generate 6GB of traffic for SFCs of lengths 4 to 30, and send it to the classifier at rates of 2MB/s, 10MB/s, and 50MB/s. The process is repeated 10 times.
We 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 (NSH and Segment Routing). In all scenarios, we assume that the IBN at the end of the first domain strips off the SFC header, and adds a MPLS label that can be used by the second domain's ingress IBN in order to identify the global SFC as well as the sub-SFCs position, and insert the local domain's corresponding headers, before sending the packet through the VXLAN tunnel associated with the IBN of the second domain. Packets from SFCs of lengths 4 to 30 are generated and sent at rates of 2MB/s, 10MB/s, and 50MB/s. Each SFC can be identified by the classifier using the packet headers and payload.

. Figures 11, 12, and 13 feature the measured latency in milliseconds as well as the 95% Confidence Interval for each SFC number and packet rate. We can observe that the end-to-end latencies range between *12ms* and *125ms* depending on the encapsulation type and SFC length, with slight variations related to the number of SFCs and packet rate. Indeed, as SFC length increases, it can be noticed that the full NSH scenario exhibits the lowest latency values, with around *12-14ms* for the shorter SFCs, and up to *61ms* for the longer ones regardless of traffic rates and SFC number, with little increase in latency between successive SFC lengths. In contrast, the latency values for the full Segment routing and hybrid scenarios increase at higher rates, reaching *125ms*, and *109ms* respectively.

. This behavior can be explained by the total size of headers. Indeed, the NSH header is fixed regardless of SFC length with 24 bytes for MD-Type 1, 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; meaning that the total packet size using the Segment Routing encapsulation becomes larger as SFC length increases. It can also be observed here that the number of SFC also affects the difference in latency for the longer SFCs ($\geq$ 20 SFs), since the latency values for the hybrid scenario are closer to the ones obtained with the Segment Routing scenario when the number of SFCs increases. Indeed, the gap between the hybrid and Segment Routing scenario's latency values gradually decreases from *12-17ms* for 50 SFCs, to *11-15ms* for 150 SFCs, and only *2-5ms* for 450 SFCs. In contrast, for the shorter SFCs ($<$ 20 SFs), similar latency values can be observed across traffic rates for SFC numbers of 50 and 150. For

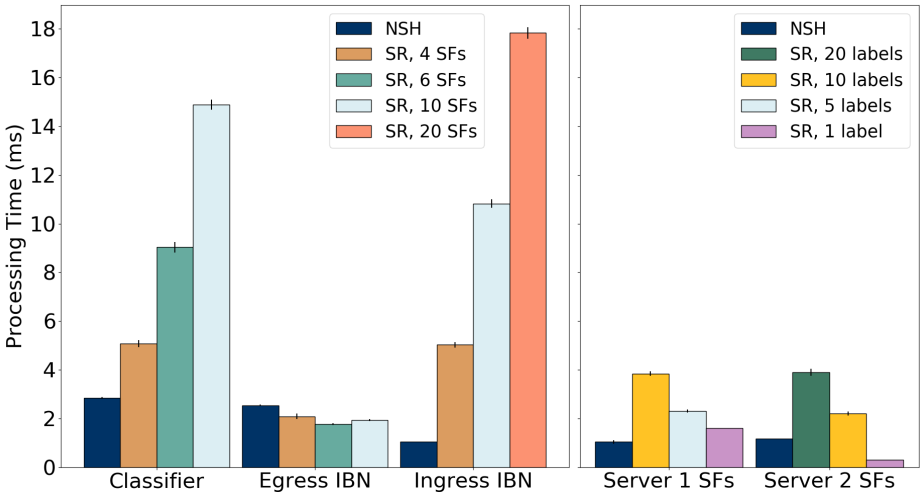


Figure 14: Processing Time per SFC Component

SFC number 450, the hybrid scenario latency values exceed the ones obtained using the Segment Routing encapsulation with up to *5ms*, as for shorter SFCs, the NSH header is still larger than the Segment Routing one.

4.4. Packet Processing Time

In addition to the end-to-end latency, we also measured during the previous experiment the processing time by each component of the SFC from the reception of a packet until the emission of the new encapsulated packet. This allows us to determine the latency added by the packet processing operations that are necessary in order to implement both encapsulation types. Figure 14 features the mean and *95%* C.I of packet processing time for the classifier, service functions and both ingress and egress IBNs of both domains,for both encapsulation types regardless of the number of SFCs and packet rate; these two factors did not affect the results. For the classifier and egress IBN of the first domain, the figure features the processing times for the NSH and Segment Routing encapsulations when the sub-SFC is composed of 4, 6, and 10 SFs. For the ingress IBN on the second domain, the values are provided for the NSH encapsulation, and Segment Routing when SFC length is of 4,10, and 20 SFs. For the Service Functions, the values are provided for NSH, and Segment Routing when the received packets have 20, 10, 5, or one label, which matches the first, middle, and last position of the longest sub-SFC in each domain.

. The figure shows that the processing times for the Service Functions are similar for both domains, where the mean processing time for the NSH encapsulation reaches *1.03ms* and *1.16ms* respectively. For Segment Routing, the processing time decreases as the number of labels decreases, with values that range between

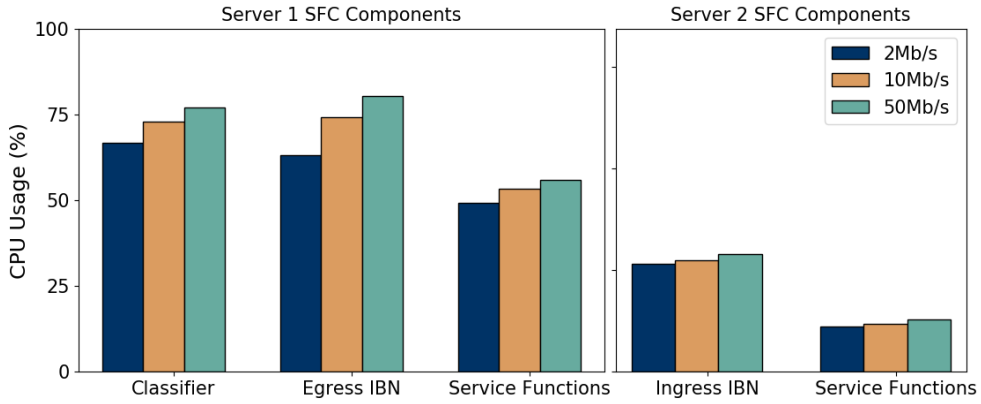
3.83ms and *0.29ms* for the first and last SF respectively. Note that the difference in processing times between the domains is due to the different hardware configurations of the servers as shown in Table 1. The same observation can be made for the classifier and ingress IBN as the mean processing time using Segment Routing increases with SFC length. The processing time reaches the value of *17.84ms* when 20 labels are added by the ingress IBN as opposed to *5ms* for 4 labels. As for the NSH encapsulation, the process time remains minimal with *2.84ms* and *1.03ms* for the classifier and ingress IBN, respectively.

. This difference can be imputed to the encapsulation process performed by both components, which differs depending on the encapsulation type. Indeed, once the sub-SFC is identified by the classifier and ingress IBN, if the NSH encapsulation is enforced, the fixed-length header is added, and the packet is directly sent to the first SF in the chain. However, for the Segment Routing encapsulation, the ID of each SF in the SFP is encoded into an individual MPLS label, meaning that for each SF in the chain, a 4-byte MPLS header is stacked over the original packet. Therefore, the longer the SFC is, the longer it takes for the classifier and ingress IBN to construct the MPLS encapsulation of the packet before sending it to the next Service Function. Finally, the egress IBN exhibits similar results for all SFC lengths in Segment Routing with a processing time of *1.76-2.08ms*, which is due to the fact that the packets reaching the IBN always have only one label left, regardless of SFC length. For the NSH encapsulation, similar to the classifier, the processing time reaches *2.54ms*.

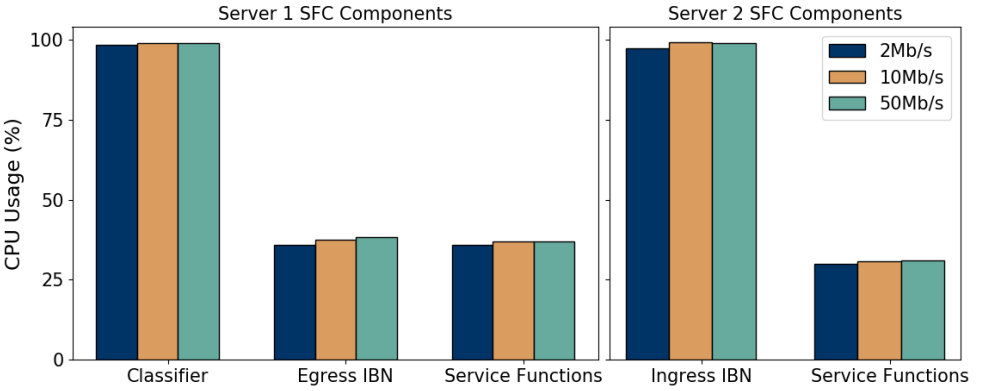
. Another observation that can be made from comparing these results to the end-to-end latency is that the processing time of a single Service Function represents *1-10%* of the total latency when the ingress IBN's processing time represents *8-14%* of the total time for Segment Routing, and *1-10%* for NSH. Depending on the encapsulation type, the classifier processing time makes up for *5-25%* of the total time for NSH, while it represents *12-40%* of the total time for Segment Routing, depending on SFC length. As for the ingress IBN's processing time, it makes up for *4-20%* of the end-to-end latency using NSH and *2-16%* using Segment Routing.

4.5. Processing Load

For this last metric, we probe the CPU load for each of the SFC components in order to determine the CPU consumption of the classification, encapsulation, and decapsulation operations for both NSH and Segment Routing headers. Figures 15a and 15b illustrate the CPU loads for the different components, for all of the simultaneously deployed SFCs for NSH and Segment Routing respectively, while Figures 16a and 16b show the CPU consumption values for an SFC comprised of 30 SFs for NSH and Segment Routing respectively, where 10 SFs are deployed on the first domain, and the remaining 20 SFs are deployed on the second domain. In all figures, the first part gathers the mean CPU consumption values of the SFC components deployed on the first domain, namely, the classifier, egress IBN, and SFs. While the second part comprises the remaining



(a) Encapsulation: Network Service Headers



(b) Encapsulation: Segment Routing

Figure 15: CPU Usage of the SFC components for all rates and all SFCs

SFs, and the ingress IBN that are deployed on the second domain. We measure the CPU values for the traffic rates of 2MB/s, 10MB/s, and 50MB/s.

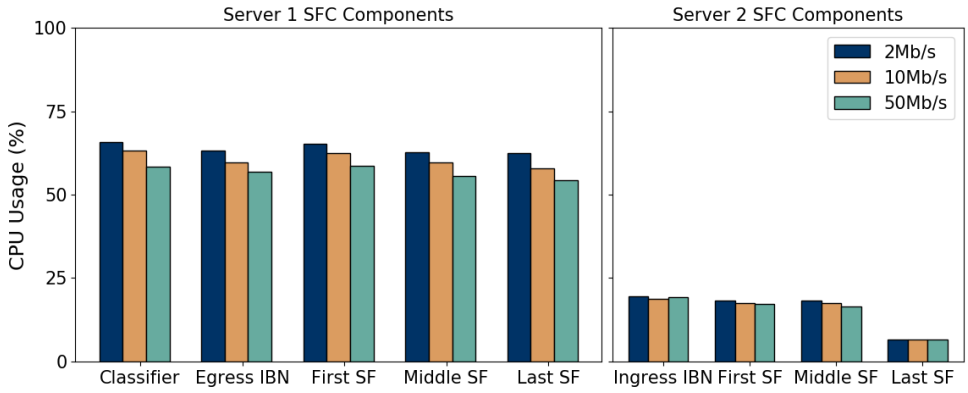
. In all figures, and for all of the components and encapsulations, we can notice a slight increase in the CPU load when the traffic rate increases. It can also be observed that for the NSH encapsulation, the CPU load remains stable across components, while it fluctuates for Segment Routing depending on the component, its function, and its position along the SFC. Looking at the CPU values for all SFCs in Figure 15, we can determine that for NSH, the classifier and egress IBN are the components that consume the most CPU, with values between *53%* and *84%*, while the ingress IBN consumes *26-28%* of CPU, and the Service Functions consume *11-66%* depending on the deployment server. For Segment Routing, the CPU consumption for Service Functions are similar for both servers, with values approximating *30%* despite the difference in resources between servers, which can be imputed to the fact that longer SFCs are deployed on the second server. The egress IBN also consumes around *30%* of CPU, while the classifier and ingress IBN consume the most resources with values of *92-99%*, which is due to the process of adding the MPLS labels to the packets at the entry of the domains.

Next, we consider the CPU usage results for the SFC comprising 30 VNFs, as shown in Figure 16. For the NSH encapsulation, it can be seen that the CPU load for all of the components of each server are similar, with values of *57-65%* for the first server, and *17-19%* for the second server, except for the last SF of the SFC, which only receives packets, consuming *5-6%* of CPU. For the Segment Routing encapsulation, the CPU usage is at its peak at the entry of each domain, with values that exceed *90%* for both the classifier, and the ingress IBN. Then, the CPU usage gradually decreases from *60%* in the first SF, to *40%* in the middle SF, then finally *28%* in the last SF and egress IBN of the first domain, and *6%* in the last SF of the second domain. These values correlate to the size of the packets that each Service Function needs to process, as for each Service Function that a packet passes through, the outer MPLS header is stripped off, thus decreasing the size of packets as they approach the end of the sub-SFC as opposed to the NSH encapsulation, where the size of the headers remains fixed while the value of the SI is decreased.

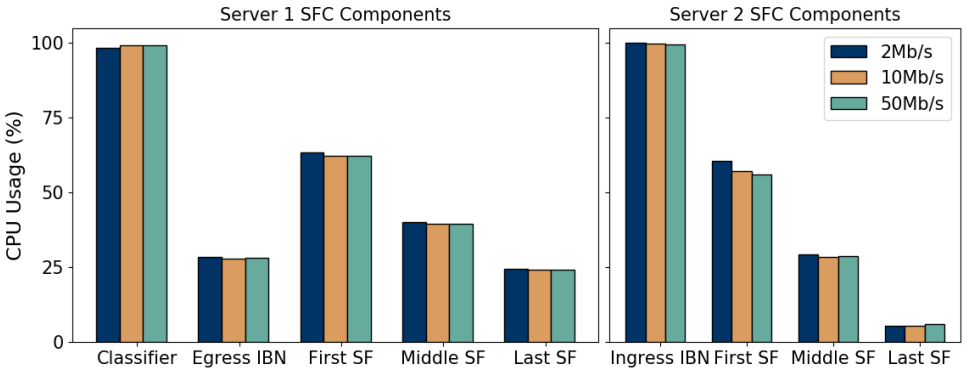
. Note that these values can change depending on the packet rate, as well as the software and hardware configuration. They cannot be generalized to other larger scale deployments as they may not apply.

4.6. Results Discussion

By synthesizing the obtained results, it can be concluded that the workflow of our proposed architecture enables an end-to-end deployment across domains with minimal time compared to the time consumed by VNF deployment at the VIM level. It is dependent on the hardware configuration of the infrastructure and the virtualization technology deployed by the operator. Moreover, as seen in section 4.2, the orchestration and network rule deployment times remain



(a) Encapsulation: Network Service Headers



(b) Encapsulation: Segment Routing

Figure 16: CPU Usage of the SFC components for all rates, for SFC of 30 VNFs

stable as SFC length increases, which proves the scalability of our architecture regarding SFC length. As for the scalability related to the number of domains, since the deployment process is performed by each local orchestrator in a parallel manner, the number of domains would not have an effect on the end-to-end deployment time. The latter would only be affected by the deployment time on the least efficient domain, since the MDO waits for the confirmation messages of all of the local orchestrators before concluding the SFC deployment process.

Comparing the different results of the NSH and Segment Routing encapsulations, and looking at the end-to-end latency in section 4.3, it can be noticed that the values for the full Segment Routing and the hybrid scenarios increase at a higher pace than those of the full NSH scenario as the SFC length increases regardless of packet rate. Furthermore, by analyzing the individual processing times of the different SFC components in section 4.4, we can perceive that the values for NSH remain stable across components, while they get multiplied by a factor of 5 to 17 for Segment Routing on the classifier and ingress IBN, depending on SFC length. Both of these differences are due to the difference in size between the Network Service Header and the Segment Routing header; the latter getting larger as the SFC increases in size. This gap is at its peak value at the ingress of a domain, as the Segment Routing encapsulation size decreases along the chain. This explains why the difference in time is observed at the classifier and ingress IBN level.

. Therefore, the choice of the implemented encapsulation type can be made based on the typical length of the sub-SFC that ought to be deployed by each domain: a shorter sub-SFC may benefit from the Segment Routing encapsulation as the total encapsulation size would remain minimal, thus ensuring a lower overhead. However, the NSH encapsulation should be considered for longer SFCs since the latter encapsulation type is not scalable. Indeed, the header size for Segment Routing increases proportionally to SFC size, thus multiplying the encapsulation times at both the classifier, and ingress IBNs of each domain, and increasing the end-to-end latency compared to NSH, as well as packet overhead which in turn would consume higher bandwidths. Additionally, as the header size increases, the total packet size might exceed the Maximum Transfer Unit (MTU), causing fragmentation issues. In terms of CPU usage, the results detailed in section 4.5 are consistent with the previous metrics, where the CPU usage for NSH is similar for the SFC components, with relatively low values, while for Segment Routing, the CPU usage decreases as packets are forwarded along the SFC, with higher consumption values at the entry of the domains, and minimal values at the end of the chain. Therefore, more CPU resources should be allocated to the SFC components at the entry of the SFC. However, more testing may be required in order to study the CPU usage of these components in larger-scale networks.

5. Open Issues

In this paper, we proposed an architecture for end-to-end cross-domain SFC implementation. We detailed the instantiation process workflow, as well as the end-to-end packet forwarding mechanism. However, some challenges related to cross-domain SFC implementation remain untackled. In this section, we outline the identified research issues that are still open, and give a brief overview of the existing research contributions.

5.1. End-to-End Life-cycle Management

In this work, we proposed an instantiation workflow for multi-domain SFC deployment, but regardless of the level of control that the owner disposes of (monitoring only, limited control, or full control), the SFC owner should be able to perform life-cycle management operations on its service chain at runtime. The Or-Or reference point provides interfaces that support basic Network Service life-cycle management operations, but SFC life-cycle management requires support of additional operations such as the addition, deletion, or reordering of service functions. Furthermore, if one of the aforementioned operations leads to the addition, or deletion, or the reordering of complete sub-chains, which would alter the higher level Service Function Path, a mechanism should be implemented in order to allow the re-configuration of the local domain IBNs.

5.2. Inter-domain Packet Forwarding

As previously stated, if we suppose that the WAN domain is not SFC aware, each IBN needs to directly send the packets at the end of its sub-chain to the IBN of the next domain, thus each IBN is required to dispose of information on how to reach the IBN of the next domain for each sub-chain that has been deployed. This can be achieved either by means of a discovery protocol that would be run by the IBNs in order to identify the IBNs of the other domains, or by obtaining that information from the multi-domain orchestrator that would have to keep track of the address of the IBN of each domain. Furthermore, when deploying sub-SFCs, each pair of consecutive IBNs should also run a negotiation protocol in order to determine the communication protocol that would be used in order to exchange SFC packets, as well as the means to identify the higher-level SFC that the packets belong to.

5.3. Encapsulation Mapping

If we consider the scenario where the WAN domain is SFC-aware, the IBN strips off the higher level encapsulation of packets at each domain's entry and replaces it by the domain's encapsulation for the identified sub-chain; the IBN should also be able to retrieve and restore that same higher level encapsulation at the egress of the domain. Note that the higher-level encapsulation might contain specific metadata that could not be restored by a simple re-classification of the packets by the IBN. The hierarchical SFC document by the IETF [30] lists a few methods that could be envisaged to retain these metadata such as

header nesting, flow state saving, or pushing the upper level encapsulation into the metadata field of the lower level encapsulation. However, at the time of this writing, these methods have not yet been evaluated, and new ones could also be proposed by future research works.

5.4. Security

As with any service, the confidentiality and integrity of data as well as the SFC's availability must be ensured. To this end, at the entry of the SFC, security functions should inspect the entering traffic in order to avoid attacks and intrusions. Then, each SFC component and link should be secured, as a compromised SF or SFF would allow an attacker to access and/or change the content of packets, or change the forwarding path of packets through packet header or forwarding rule modification. This task is more challenging in a multi-domain scenario, due to the independent management of each domain, as well as the inter-domain forwarding part. Therefore, each local domain should guarantee the confidentiality and integrity of the SFC data that is processed by its sub-SFC, and allow the implementation of end-to-end consistency verification measures. Furthermore, since the SFC packets are forwarded between the domains through untrusted networks, encrypted tunnels should be set between the IBNs in order to ensure the confidentiality and integrity of the packets. Finally, at the orchestration level, an authentication procedure should be run between the orchestrators of each domain and the MDO, and the communication channel should be secured.

6. Conclusion

In this paper, we introduced a novel architecture that enables multi-domain SFCs 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 also detailed the SFC instantiation process, as well as the partitioning of the NSDs in order to perform the deployment of the sub-chains on each local domain. We devised on an SDN-based technology-agnostic end-to-end packet forwarding mechanism that ensures cross-domain compatibility by adding interfacing components, and assessed its effectiveness with a Proof of Concept implementation that has been evaluated by measuring different Key Performance Indicators for SFCs. The obtained results demonstrate our architecture's scalability and efficiency, and allows us to draw conclusions related to the most suitable encapsulation type depending on SFC length. Finally, we provides an analysis of the remaining open issues for end-to-end multi-domain SFC implementation and management.

References

[1] ETSI, MEC in 5G networks, White Paper (WP) 28, ETSI (06 2018).
URL `https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf`

[2] L. Qu, M. Khabbaz, C. Assi, Reliability-aware service chaining in carrier-grade softwarized networks, IEEE Journal on Selected Areas in Communications (2018) 1–1.

[3] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, P. Suter, Serverless Computing: Current Trends and Open Problems, Springer Singapore, Singapore, 2017, pp. 1–20. doi:10.1007/978-981-10-5026-8_1.

[4] J. M. Halpern, C. Pignataro, Service Function Chaining (SFC) Architecture, RFC 7665 (Oct. 2015). doi:10.17487/RFC7665.
URL https://rfc-editor.org/rfc/rfc7665.txt

[5] ETSI GS NFV-MAN 001, Network Functions Virtualisation (NFV); Management and Orchestration , Tech. rep., ETSI (December 2014).
URL http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[6] X. Zhong, Y. Wang, X. Qiu, Service function chain orchestration across multiple clouds, China Communications 15 (10) (2018) 99–116. doi:10.1109/CC.2018.8485473.

[7] G. Sun, Y. Li, D. Liao, V. Chang, Service function chain orchestration across multiple domains: A full mesh aggregation approach, IEEE Transactions on Network and Service Management 15 (3) (2018) 1175–1191. doi:10.1109/TNSM.2018.2861717.

[8] D. Dietrich, A. Abujoda, A. Rizk, P. Papadimitriou, Multi-provider service chain embedding with nestor, IEEE Transactions on Network and Service Management 14 (1) (2017) 91–105.

[9] A. Abujoda, P. Papadimitriou, Distnse: Distributed network service embedding across multiple providers, in: 2016 8th International Conference on Communication Systems and Networks (COMSNETS), 2016, pp. 1–8. doi:10.1109/COMSNETS.2016.7439948.

[10] J. Zu, G. Hu, Y. Wu, D. Shao, J. Yan, Resource aware chaining and adaptive capacity scaling for service function chains in distributed cloud network, IEEE Access 7 (2019) 157707–157723.

[11] J. Pei, P. Hong, K. Xue, D. Li, Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system, IEEE Transactions on Parallel and Distributed Systems 30 (10) (2019) 2179–2192.

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

[13] F. Esposito, D. Di Paola, I. Matta, On distributed virtual network embedding with guarantees, IEEE/ACM Transactions on Networking 24 (1) (2016) 569–582.

[14] B. Addis, D. Belabed, M. Bouet, S. Secci, Virtual network functions placement and routing optimization, in: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 2015, pp. 171–177.

[15] M. Mechtri, C. Ghribi, D. Zeghlache, A scalable algorithm for the placement of service function chains, IEEE Transactions on Network and Service Management 13 (3) (2016) 533–546.

[16] ETSI GS NFV-IFA 030, Network functions virtualisation (nfv) release 3; management and orchestration; multiple administrative domain aspect interfaces specification, Tech. rep., ETSI (April 2019).
URL `https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-IFA030v3.2.1-GS-MultiDomainMANO-spec.pdf`

[17] K. Katsalis, N. Nikaein, A. Edmonds, Multi-domain orchestration for nfv: Challenges and research directions, in: 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), 2016, pp. 189–195. doi:10.1109/IUCC-CSS.2016.034.

[18] T. Taleb, I. Afolabi, K. Samdanis, F. Z. Yousaf, On multi-domain network slicing orchestration architecture and federated resource control, IEEE Network 33 (5) (2019) 242–252.

[19] M. Mechtri, C. Ghribi, O. Soualah, D. Zeghlache, Nfv orchestration framework addressing sfc challenges, IEEE Communications Magazine 55 (6) (2017) 16–23. doi:10.1109/MCOM.2017.1601055.

[20] A. M. Medhat, G. A. Carella, M. Pauls, 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), 2017, pp. 327–333. doi:10.1109/NFV-SDN.2017.8169879.

[21] W. Ding, W. Qi, J. Wang, B. Chen, Openscaas: an open service chain as a service platform toward the integration of sdn and nfv, IEEE Network 29 (3) (2015) 30–35. doi:10.1109/MNET.2015.7113222.

[22] J. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, S. Sargento, Toward a telco cloud environment for service functions, IEEE Communications Magazine 53 (2) (2015) 98–106. doi:10.1109/MCOM.2015.7045397.

[23] S. Kumar, M. Tufail, S. Majee, C. Captari, S. Homma, Service Function Chaining Use Cases In Data Centers, Internet-Draft draft-ietf-sfc-dc-use-cases-06, Internet Engineering Task Force, work in Progress (Feb. 2017).

URL https://datatracker.ietf.org/doc/html/draft-ietf-sfc-dc-use-cases-06

[24] V. Mehmeri, X. Wang, Q. Zhang, P. Palacharla, T. Ikeuchi, I. T. Monroy, Optical network as a service for service function chaining across datacenters, in: 2017 Optical Fiber Communications Conference and Exhibition (OFC), 2017, pp. 1–3.

[25] P. Quinn, U. Elzur, C. Pignataro, Network Service Header (NSH), RFC 8300 (Jan. 2018). doi:10.17487/RFC8300.
URL https://rfc-editor.org/rfc/rfc8300.txt

[26] B. Martini, F. Paganelli, A. A. Mohammed, M. Gharbaoui, A. Sgambelluri, P. Castoldi, Sdn controller for context-aware data delivery in dynamic service chaining, in: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–5. doi:10.1109/NETSOFT.2015.7116146.

[27] P. B. Pawar, K. Kataoka, Segmented proactive flow rule injection for service chaining using sdn, in: 2016 IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 38–42. doi:10.1109/NETSOFT.2016.7502439.

[28] Internet-Draft draft-xu-sfc-using-mpls-spring-01, Internet Engineering Task Force, informational. [link].
URL https://tools.ietf.org/id/draft-xu-sfc-using-mpls-spring-01.html

[29] H. Hantouti, N. Benamar, T. Taleb, A. Laghrissi, Traffic steering for service function chaining, IEEE Communications Surveys Tutorials 21 (1) (2019) 487–507. doi:10.1109/COMST.2018.2862404.

[30] D. Dolson, S. Homma, D. Lopez, M. Boucadair, Hierarchical Service Function Chaining (hSFC), RFC 8459 (Sep. 2018). doi:10.17487/RFC8459.
URL https://rfc-editor.org/rfc/rfc8459.txt

[31] G. Li, G. Li, T. Li, Q. Xu, B. Feng, H. chun Zhou, Multi-domain Service Forwarding For NSH, Internet-Draft draft-li-sfc-nsh-multi-domain-04, Internet Engineering Task Force, work in Progress (Apr. 2018).
URL https://datatracker.ietf.org/doc/html/draft-li-sfc-nsh-multi-domain-04

[32] TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0. Edited by Shitao Li and John Crandall. 11 May 2017. OASIS Committee Specification Draft 04. http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.html.

[33] Linux Containers (LXC), https://linuxcontainers.org/.

[34] Open vSwitch, https://www.openvswitch.org/.

[35] Scapy Project, https://scapy.net/.

[36] Hping3, https://linux.die.net/man/8/hping3.

[37] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, R. Shakir, Segment Routing Architecture, RFC 8402 (Jul. 2018). doi:10.17487/RFC8402.
URL https://rfc-editor.org/rfc/rfc8402.txt