Sorbonne Université

École doctorale
Informatique, télécommunications et électronique de Paris

EURECOM

Étude du Service Function Chaining dans un contexte multi-domaines

Présentée par Nassima TOUMI

Thèse de doctorat en Sciences de l'Information et de la Communication

Dirigée par Adlen KSENTINI
Olivier BERNIER

Présentée et soutenue publiquement le 05 Janvier 2021

Devant un jury composé de :

Prof. Carla Fabiana Chiasserini	Politecnico di Torino	Rapporteur
Prof. Hind Castel	IMT Sud Paris	Rapporteur
Prof. Raymond Knopp	EURECOM	Examinateur
Prof. César Viho	University Rennes 1	Examinateur
Dr. Jeremie Leguay	Huawei Technologies	Examinateur
Dr. Djamal-Eddine Meddour	Orange	Invité
Prof. Adlen Ksentini	EURECOM	Directeur de Thèse
Dr. Olivier Bernier	Orange Labs	Co-Directeur de Thèse

Sorbonne University

Doctoral School of Informatics, Telecommunications and Electronics of Paris

EURECOM

Enabling Service Function Chaining in a multi-domain context

Presented by Nassima TOUMI

Dissertation for Doctor of Philosophy in Information and Communication Engineering

Directed by Adlen KSENTINI
Olivier BERNIER

Publicly presented and defended on 05 January 2021

A Jury committee composed of:

Prof. Carla Fabiana Chiasserini	Politecnico di Torino	Reviewer
Prof. Hind Castel	IMT Sud Paris	Reviewer
Prof. Raymond Knopp	EURECOM	Examiner
Prof. César Viho	University Rennes 1	Examiner
Dr. Jeremie Leguay	Huawei Technologies	Examiner
Dr. Djamal-Eddine Meddour	Orange	Invited
Prof. Adlen Ksentini	EURECOM	Thesis Director
Dr. Olivier Bernier	Orange Labs	Thesis co-Director

Abstract

Service Function Chaining (SFC) is a networking concept by which traffic is steered through a set of ordered functions composing an end-to-end service. It represents one of the facilitating technologies for 5G, and is enabled by the Network Function Virtualization (NFV) and Software Defined Networks (SDN) paradigms. Many contributions have been made to support SFC, from chain composition to its deployment and life-cycle management. However, few research works study SFC orchestration. Even fewer consider the multi-domain context, which remains an open issue with many challenges that need to be tackled to provide end-to-end services.

Indeed, in the multi-domain context, SFC placement faces new challenges related to the lack of visibility on the local domain's networks. The domain operators are often reluctant to unveil details on their topology to external parties. Furthermore, the new 5G use cases introduce new requirements related to Quality of Service (QoS) and Quality of Experience (QoE), requirements that the placement process needs to optimize simultaneously. Another issue stems from the possible heterogeneity in encapsulation methods that are implemented by each domain, which means that an interfacing entity is needed at the ingress of each domain, to add the sub-SFCs encapsulation that is implemented by the domain. Furthermore, to ensure the end-to-end forwarding of SFC packets, each domain should be able to classify the packets at its network's ingress, and be aware of the next domain in the global SFC in order to forward the packets at the end of its local sub-SFC.

In this thesis, different solutions are proposed to enable Service Function Chaining over multiple domains. First, a hierarchical framework is elaborated for SFC placement with limited visibility, and an ILP model is fomulated for the optimization problem, which aims to minimize the overall cost, and the end-to-end latency, using the weighted sum approach. A heuristic is also developed for scalability. The evaluation results show the heuristic's efficiency with a limited visibility, with solutions that are close to optimal by $89 - 96\%$ in realistic times.

Next, the formulation is improved to express the user's preferences more accurately using Physical Programming, and SLA classes are determined according to the 5G use cases and their requirements in terms of latency and bandwidth per user. This approach is compared to the previously modeled ILP. The results show that Physical Programming allows the generation of placement solutions that are closer to the user's preferences for all of the optimization objectives.

Finally, an ETSI-compliant orchestration framework is proposed, to deploy multi-domain SFCs while taking into account the difference in encapsulations implemented by each domain. The mechanisms for the end-to-end SFC deployment and deletion are detailed, and a Proof of Concept is implemented. The evaluation shows that SFC deployment times vary mainly depending on the VNF instantiation process, while the orchestration and network configuration times remain stable. The observation of the end-to-end latencies also gave insights on the type of encapsulation that should be implemented depending on SFC length.

Keywords : Service Function Chaining (SFC); Network Function Virtualization (NFV); Software Defined Networks (SDN); Multi-Domain ; Optimization; Physical Programming; Orchestration and Management (MANO)

Résumé

Les avancées technologiques récentes, ainsi que la démocratisation des smartphones et objets intelligents ont conduit à l'émergence de nouveaux cas d'usage d'internet (voitures autonomes, e-santé, maisons connectées...etc), ainsi qu'à la multiplication du nombre de terminaux connectés aux réseaux mobiles et du volume de données échangées. Les réseaux de quatrième génération (4G) n'étant pas en mesure de supporter ces nouveaux besoins, notamment en termes de bande passante et de latence, une cinquième génération de réseaux (5G) a été développée afin de pallier à ces exigences. Au niveau du coeur de réseau, elle s'appuie principalement sur deux technologies : Les Software Defined Networks (SDN), ainsi que la Network Function Virtualization (NFV).

Ces deux technologies permettent également la mise en oeuvre du Service Function Chaining (SFC). Celui-ci est un concept qui consiste en la décomposition d'un service donné en plusieurs blocs fonctionnels, appelés Service Functions (SFs), qui effectuent tour à tour des traitements spécifiques des données afin de délivrer le service final à l'utilisateur. Les données d'une SFC sont acheminées entre les différentes fonctions par des Service Function Forwarders (SFFs) dans un ordre précis, et nécessitent un mécanisme d'identification et de routage de paquets.

Plusieurs travaux ont abordé le Service Function Chaining, en proposant différentes solutions de placement, d'orchestration et de déploiement, mais la majorité s'intéresse uniquement au cas des SFCs mono-domaines. Cependant, il existe des cas où pour des besoins fonctionnels ou de sécurité, les fonctions d'une même SFC doivent être placées sur des domaines différents. Il est à noter que le terme multi-domaines peut concerner plusieurs entités administratives, mais également des départements appartenant à une même entité administrative, mais dont la gestion se fait de manière autonome et indépendante pour des raisons pratiques. Ce scénario comporte des spécificités qui nécessitent d'être prises en compte lors du déploiement de SFCs, notamment, le manque de visibilité sur l'infrastructure locale des domaines, ainsi que l'hétérogénéité des méthodes d'encapsulation et de routage des paquets entre les différents domaines.

Dans cette thèse, plusieurs solutions sont élaborées afin de permettre le déploiement de SFCs, dès l'étape de la réception d'une requête de déploiement, jusqu'à la fin de déploiement effectif. La première problématique abordée est celle du placement de SFCs avec une visibilité limitée sur les infrastructures des domaines locaux. En effet, les administrateurs de domaines sont réticents à divulguer des détails critiques sur leur infrastructure, et vont donc communiquer le moins d'informations possible. Une architecture de placement hiérarchique est proposée, où un orchestrateur centralisé reconstitue une vue abstraite du réseau à partir des informations communiquées par chaque domaine et effectue un placement initial des SFCs, puis partitionne la requête et envoie chaque sous-SFC aux domaines locaux pour qu'ils effectuent leur placement. Une modélisation linéaire (Integer Linear Programming) optimale, ainsi qu'une heuristique sont développées, avec pour objectif de minimiser le coût de déploiement et la latence de bout en bout, en utilisant une somme pondérée des valeurs normalisées comme objectif d'optimisation. Les résultats des simulations démontrent l'efficacité de l'heuristique combinée à l'architecture hiérarchique, avec des résultats se rapprochant de l'optimal à $89 - 96\%$.

Par la suite, le modèle d'optimisation hiérarchique est enrichi afin d'exprimer de manière plus précise les préferences des utilisateurs. Pour ce faire, l'approche de Physical Programming est employée afin de formuler les fonctions objectif à l'aide d'intervalles de préférences définies par l'utilisateur. Trois méthodes sont utilisées : Le Linear Physical Programming, le Non Linear Physical Programming, et le Global Physical Programming. Trois objectifs d'optimisation sont considérés : le coût, la latence de bout en bout, mais aussi la bande passante allouée par utilisateur. Des SLAs inspirées des cas d'usage de la 5G sont définies avec différentes exigences par rapport aux indicateurs de QoS, et un algorithme Branch and Bound ainsi qu'une heuristique mémétique sont développés. L'évaluation de ces algorithmes montre que les méthodes de Physical Programming permettent d'obtenir des résultats satisfaisant chaque objectif de manière individuelle, et que les solutions obtenues sont plus satisfaisantes pour les utilisateurs que celles de l'ILP.

Enfin, une solution d'orchestration est conçue et mise en oeuvre afin de permettre le déploiement des SFCs multi-domaines, en effectuant la configuration de l'acheminement des paquets de bout en bout, et en assurant une traduction des en-têtes d'encapsulation entre les domaines. Une architecture d'orchestration multi-domaines conforme au standard ETSI-MANO est détaillée, avec l'incorporation d'un composant d'interfaçage permettant d'insérer l'encapsulation de chaque sous-SFC à l'entrée du domaine, et d'assurer le routage des paquets entre domaines. Un mécanisme de déploiement de SFCs est également mis en place afin de permettre l'échange d'informations nécessaires entre composants pour le routage des paquets de bout en bout. Une preuve de concept est développée afin de valider cette architecture, en implémentant une plateforme de déploiement de

SFCs sur deux machines physiques. L'évaluation des mécanismes de déploiement et de terminaison démontre la scalabilité de cette solution en termes de temps d'orchestration et de configuration qui restent stables avec l'augmentation de la longueur des SFCs. Par ailleurs, les mesures de latence, en employant deux types d'encapsulation différents (NSH et Segment Routing), permettent d'observer des différences entre les deux encapsulations au niveau de l'entrée des domaines, qui peut être expliquée par la taille des en-têtes ajoutés, et des latences de bout en bout qui varient de 12 à $120ms$ selon la longueur de SFC.

Mots-clé : Chaînes de service (SFC); Virtualisation des fonctions réseau (NFV); Software Defined Networks (SDN); Multi-Domaines ; Optimisation; Physical Programming; Orchestration et Management (MANO)

Remerciements

Ce manuscrit est le fruit de trois années de travail. Je tiens à exprimer ma reconnaissance à toute personne ayant contribué de près ou de loin à l'aboutissement de ce projet.

Mes premiers remerciements s'adressent à mon Directeur de thèse, *Pr. Adlen Ksentini*, pour m'avoir fait confiance et donné l'opportunité de travailler sur cette thèse. Je lui suis reconnaissante pour son implication, la qualité de son encadrement, et ses remarques et conseils qui ont permis d'orienter mes réflexions tout au long de ces trois années.

Je remercie également mes encadrants industriels à Orange, *Mr. Olivier Bernier* et *Dr. Djamal-Eddine Meddour* pour leur disponibilité, et pour m'avoir fait bénéficier de leurs connaissances et expertise. Leurs conseils et orientations m'ont été d'une grande aide durant cette période.

Mes respects et remerciements à *Pr. Carla Fabiana Chiasserini* et *Pr. Hind Castel* pour avoir accepté d'être rapporteuses de ma thèse. Mes remerciements vont également aux membres du jury *Pr. César Viho*, *Dr. Jeremie Leguay*, et en particulier à *Pr. Raymond Knopp* pour avoir présidé ce jury lors de ma soutenance. Par ailleurs, j'assure ma reconnaissance aux membres de mon comité de suivi de thèse, *Dr. Géraldine Texier* et *Pr. Navid Nikaein* pour leur accompagnement et leurs recommandations.

Je tiens à remercier chaleureusement mes collègues de l'équipe *AMOX* à Orange, pour leur bienveillance et bonne humeur, en particulier *El Hadja Chaalal*, *Corentin Eon*, *Stéphan Del Burgo*, *Eric Le Jamtel*, et *Olivier Montanuy*, ainsi que mon chef d'équipe *Pierre Henry* pour son appui. Par ailleurs, je souhaite remercier *Mme. Sophie Salmon* et *Mme. Régine Angoujard* pour leur aide précieuse lors de mes démarches administratives, et *Dr. Séverine Jacquet* et *Mme. Thao Lang* de l'ABG pour leur accompagnement et conseils.

Je témoigne toute ma gratitude à mes Professeurs et encadrants durant tout mon parcours scolaire et universitaire, et en particulier *Dr. Chafika Benzaid*, *Pr. Abdelkader Belkhir*, *Dr. Miloud Bagaa*, et *Pr. Diego Dutra*, qui ont grandement contibué à ma formation, et m'ont initié à la recherche scientifique. Je les remercie pour leur temps, leurs efforts et conseils.

Enfin, mes derniers remerciements vont à ma famille et mes proches, et en particulier mes parents, à qui je dédie ce travail. Je leur suis reconnaissante pour leur soutien indéfectible et inconditionnel, et tous leurs efforts et encouragements pour m'aider à réaliser mes objectifs et concrétiser mes projets. À ma tante, mon frère, et mes amis : *Mohamed*, *Lynda*, *El Hadja*, *Nawel* et *Oussama* pour leur soutien et encouragements durant toute cette période.

À toutes ces personnes, je présente mes remerciements, mon respect et ma gratitude.

Table of Contents

Acronyms

3GPP 3rd Generation Partnership Project.

5GPPP 5th Generation Private Public Partnership.

API Application Programming Interface.

BSS Business System Support functions.

CAPEX Capital Expenditures.

CP Connection Point.

DM Decision Maker.

DPI Deep Packet Inspection.

ETSI European Telecommunications Standards Institute.

GPP Global Physical Programming.

IaaS Infrastructure as a Service.

IBN Internal Boundary Node.

IETF Internet Engineering Task Force.

ILP Integer Linear Programming.

IoT Internet of Things.

ITU International Telecommunication Union.

JSON JavaScript Object Notation.

KPI Key Performance Indicator.

LPP Linear Physical Programming.

MAC Media Access Control.

MANO Management and Orchestration.

MDO Multi Domain Orchestrator.

MEC Mobile Edge Cloud.

MOO Multi Objective Optimization.

MPLS Multi Protocol Label Switching.

NAT Network Address Translation.

NFP Network Function Provider.

NFV Network Function Virtualization.

NFVI Network Function Virtualization Infrastructure.

NFVO Network Function Virtualization Orchestrator.

NGMN Next Generation Mobile Networks.

NLPP Non Linear Physical Programming.

NS Network Service.

NSD Network Service Descriptor.

NSH Network Service Header.

ONF Open Networking Foundation.

OPEX Operational Expenditures.

OSS Operation System Support.

PaaS Platform as a Service.

PNF Physical Network Function.

QoS Quality of Service.

SaaS Software as a Service.

SDN Software Defined Networking.

SDO Standard Development Organization.

SF Service Function.

SFC Service Function Chaining.

SFF Service Function Forwarder.

SFP Service Function Path.

SI Service Index.

SLA Service Level Agreement.

SPI Service Path Identifier.

SR Segment Routing.

VIM Virtualized Infrastructure Manager.

VL Virtual Link.

VNF Virtual Network Function.

VNFD Virtual Network Function Descriptor.

VNFFG Virtual Network Function Forwarding Graph.

VNFM Virtual Network Function Manager.

WAN Wide Area Network.

List of Figures

List of Tables

Chapter 1

Introduction

The recent technological advances in networks and applications have introduced a set of new heterogeneous services and interactive applications with specific requirements. Further, the advent of the Internet of Things (IoT) and Massive Machine type Communication have exponentially multiplied the number of users, and traffic volumes that networks need to handle. In this context, a new generation of mobile networks technology (5G) is under development aiming to accommodate the requirements of the new use cases that have been introduced [1]. Multiple Standard Development Organizations (SDO) and partnerships such as ITU (International Telecommunication Union), 3GPP (Third Generation Partnership Project), and NGMN (Next Generation Mobile Networks Alliance) have launched initiatives such as the 5GPPP (5G Infrastructure Public Private Partnership) in order to standardize the 5G architecture, and identify its use cases and requirements [2]. In summary, the upcoming 5G networks are set to provide higher data rates (100 times the usual data rates), ultra-low latency ($< 1ms$ for radio latency, and $< 10ms$ for the end to end latency), improved connectivity to support higher user densities and traffic volumes, a "*5-nines*" reliability, support for mobility of users, while keeping cost and energy consumption at a minimum [3]. Figure 1.1 summarizes the main requirements for 5G, as defined by the 5GPPP [4]. Note that the requirements for each Key Performance Indicator (KPI) depend on the use case. Indeed, the Augmented Reality use case would require lower latencies than those needed for a classic web browsing usage. Therefore, a differentiation between the use cases and their specific requirements needed to be made. The different SDOs define three main usage categories for their use case definitions [3]:

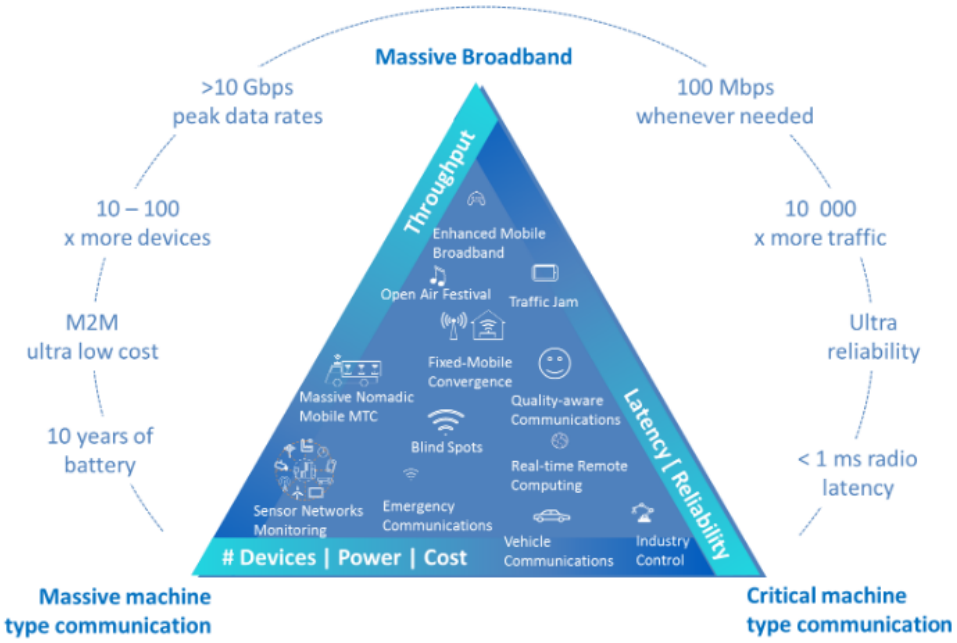


Figure 1.1: 5G NORMA Use Cases and Requirements [4]

- **Enhanced Mobile BroadBand (eMBB)**: Covers the classical internet usage scenarios, where significantly higher data rates are provided to enable applications such as the streaming of 4K and 8K UHD videos.
- **Massive Machine Type Communications (mMTC)**: This category encompasses the use cases that include machine-to-machine (M2M) communications, and require support for a very high number of connected devices and multiplied traffic volumes. Examples of use cases include IoT based applications such as smart homes, smart cities, and smart factories.
- **Ultra Reliable Low Latency Communications (URLLC)**: The use cases of this category require ultra-reliable and low-latency network links for critical communications. Examples include Autonomous Vehicles with V2X communication, drone communications, and industrial control and automation.

The 3GPP defines multiple usage scenarios for 5G, with different KPI targets [1,5]. The NGMN identifies multiple use cases, as well as user experience and system performance KPIs such as

data rates, latency, mobility, and traffic density; and describes 5G design principles and components [6, 7]. Similarly, the ITU defines key parameters and their target values for International Mobile Telecommunications (IMT) for 2020 and beyond [8]. While 5GPPP proposes a layered functional architecture for 5G dubbed 5G NORMA [9] [4], and several use cases and scenarios, with functional and performance requirements.

In order to satisfy the aforementioned constraints, two main enabling technologies have been retained for the core part of 5G: Network Function Virtualization (NFV), and Software Defined Networking (SDN). Software Defined Networking states that the control plane is decoupled from the data plane in networks: the traffic steering rules are determined by a logically centralized controller and are communicated to the forwarding network components in the data plane through a southbound interface, thus making the network dynamically programmable [10]. On the other hand, cloud computing and virtualization allowed for the rise of the Network Function Virtualization paradigm, where network functions are deployed as virtual instances on commodity servers instead of proprietary hardware appliances, therefore reducing Capital (CAPEX) and Operational (OPEX) Expenditures, and improving service flexibility and manageability [11, 12]. Along with NFV emerged the concept of micro-services, which is the decomposition of services in multiple small blocks that perform simple functions. This approach improves service modularity and resilience, adds flexibility in service management, and allows for function sharing between services [13]. However, decomposing services into a set of functions creates the need for steering traffic between those functions in a particular order so that services are properly delivered; this process is referred to as Service Function Chaining (SFC).

1.1 Context And Motivations

Service Function Chaining is a networking concept by which traffic is steered through a set of ordered functions composing an end-to-end service. Many research works explored SFC under different aspects: composition, resource allocation, as well as placement and chaining, and proposed solutions to enable and support SFCs. However, most of these works assume that the SFC is deployed on a single domain and ignore the multi-domain scenario, which remains an open issue with many challenges that need to be tackled in order to provide end-to-end services.

The multi-domain scenario occurs when the SFC requires the deployment of its components on different domains for multiple reasons such as resource shortage, the mobility of users, security, or service composition using functions that are delivered by different operators. 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 its own networking protocols. Therefore, the multi-domain context can be applied both to multiple administrative entities and to separate divisions of the same entity.

1.1.1 Multi-Domain SFC Use Cases

In this section, we present four use-cases demonstrating the need of multi-domain SFCs. The first use-case is the IoT use case where security and data aggregation functions are deployed on edge clouds; the second one is related to user mobility, where a part of the SFC is migrated/re-instantiated on a different domain post-deployment as a result of a user's change of position; the third one is related to the more generic scenario of service composition, where the end-to-end SFC is constructed by deploying and chaining functions on different domains; and the final use case refers to resource shortage, where the operator doesn't dispose of sufficient resources in order to accommodate the SFC, whether during the initial deployment, or post-deployment.

Industrial Internet of Things (IIoT) Use Case

As previously stated, the upcoming 5G networks are set to enable three main usage scenarios: enhanced mobile broadband, ultra-reliable and low-latency communications, and massive machine type communications. The latter supports the deployment of a massive number of IoT devices for several applications. Industrial Internet of Things (IIoT) represents one of the most significant applications of IoT, with use cases such as smart cities and smart factories. These use cases rely on the deployment and the collection of data from a large set of IoT devices, e.g. surveillance cameras, sensors, etc. [14]. Considering the volume of data that is generated by the different devices, the operator could benefit from placing data aggregation functions at the edge of the network to reduce the amount of consumed network resources [15]. Multiple network providers and vendors have developed commercial solutions that leverage on IoT and edge computing, such as Microsoft with Azure IoT Edge [16], or Amazon with AWS IoT Greengrass [17]. Both solutions allow the deployment of pre-processing functions at the edge of the network, which collect the data generated by the IoT devices and perform filtering and compression before sending it to the core cloud for further processing.

Furthermore, for security concerns, it is essential to protect the infrastructure from different attacks, especially since IoT devices are more vulnerable to hacking and can be used as vectors for large scale attacks [18]. The operator would need to deploy security functions that perform traffic inspection and attack detection in order to stop/mitigate attacks such as DDos attacks before they

enter the network. Therefore, security functions should also be deployed as close as possible to the sources of data.

However, due to the number and geographical distribution of the devices, the operator may not be able to afford the cost of deploying edge clouds covering the whole sensing area; instead, a less costly alternative is to deploy the security and compression functions on external edge clouds closer to the devices. In that case, as illustrated in Figure 1.2, the operator deploys a multi-domain SFC, where the data flow generated by the IoT devices is directed to the security functions deployed on the external edge clouds. After analysis, the flows that are deemed secure are transmitted to the aggregation functions also deployed on the external edge clouds, before sending the compressed data to the remaining functions of the SFC that are deployed on the operator's domain.

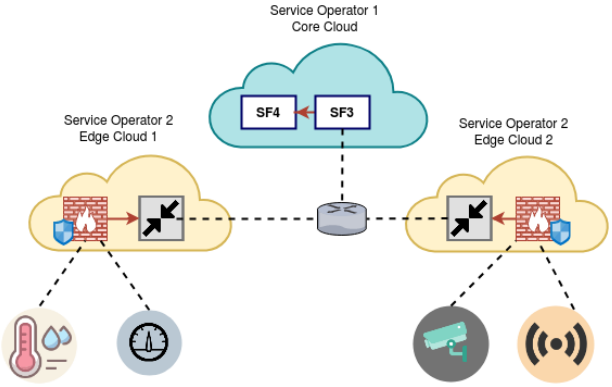


Figure 1.2: IoT Use Case: Security and data aggregation functions at the edge of the network

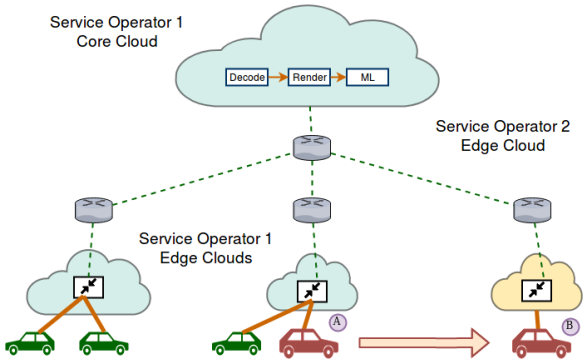


Figure 1.3: User Mobility Use Case: Autonomous Driving

User Mobility Use Case: Autonomous Driving

Autonomous driving is one of the main use cases defined for the upcoming 5G networks [19]. It leverages Vehicle to Infrastructure (V2I) as well as Vehicle to Vehicle (V2V) communication in order to collect information on the vehicles' surroundings and make appropriate driving decisions. We will consider the Vehicle to Infrastructure communication case: The cars capture information on their environment using onboarded cameras and sensors, and send them to the core cloud for treatment. The analysis function then makes decisions according to the obtained information and sends back a navigation control instruction to the vehicle. An example SFC for this use case would be as follows [20]:

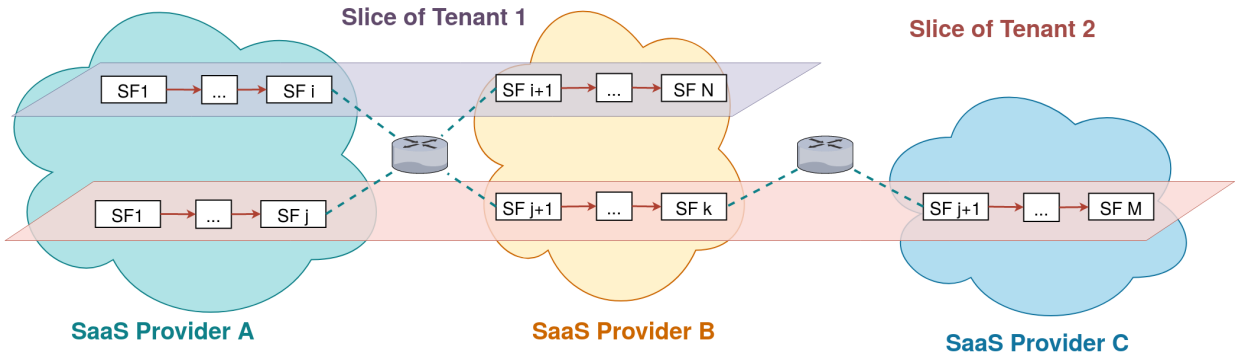


Figure 1.4: Service Composition Across Multiple Operator Domains

The flow goes first through security functions (IDS, firewall) in order to ensure that the traffic is legitimate and doesn't violate any security policy. It is then transmitted to a service prioritization function that determines that this traffic has a high priority due to the criticality of the service. After that the flow is routed to a video decoding function, then to a screen rendering function, and finally to a machine learning function in order to obtain optimal navigation control procedures to be sent to the vehicle for execution. For the sake of simplicity, the security and flow prioritization functions will be omitted in the illustration.

In this context, and similar to the previous use case, the operator deploys image compression functions at the edge of the network, therefore reducing the amount of consumed network resources. However, as illustrated in Figure 1.3 the vehicles are in constant mobility. Some cars could move far away from the operator's edge clouds, which would induce additional latency that wouldn't be acceptable for such a critical application. Indeed, a delayed response would lead to increased accident risk. Note that due to the unpredictability of the users' mobility patterns, it would be difficult for the operator to deploy edge clouds that would cover all the geographical areas visited by its users in a cost-efficient manner. A solution for the operator would then be to either deploy a new compression function or migrate an existing one to the edge cloud of a second operator closer to the vehicle, then send the traffic back to the original SFC, therefore creating a multi-domain SFC.

Service Composition

With the advent of virtualization, different cloud service models emerged: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These models enable service providers to grant access to resources, development environment, and software hosted on their infrastructure respectively, thus enabling a multi-tenant multi-layer architecture where physical and virtual resources are shared between different customers (tenants) as independent and isolated slices. In this scenario, as illustrated in Figure 1.4, the service *tenant* does not own any infrastructure and composes its own service from a set of *micro-services* where each type of function is hosted by a different cloud service provider.

As an example, the European Telecommunications Standards Institute (ETSI) Mobile Edge Cloud (MEC) in 5G white paper describes a use case where Mobile Network Operators (MNOs) buy edge cloud services from third party providers in order to extend their network, without buying edge infrastructure [15]. Furthermore, multiple companies such as Google [21], Amazon [22], Microsoft [23] or IBM [24] provide Functions-as-a-Service (FaaS), also referred to as Serverless computing, where the client deploys its own applications on cloud platforms provided and managed by the FaaS operator [25]. In order to better take advantage of the different proposed features and pricing of each vendor, the client might benefit from deploying parts of their application on the clouds of different vendors [26]. The Virtual Network Functions (VNFs) are chained together creating the SFC of the tenant, and each tenant then disposes of its own overlay slice spanning multiple domains. Note that a slice can host more than one SFC.

Resource Shortage

Initial Deployment Supposing a user requests a certain service that requires the deployment of a new SFC, but the domain operator does not dispose of sufficient resources in order to host the whole SFC; for security reasons, the operator then chooses to deploy locally the critical functions that should not be hosted on an external operator's domain (e.g. functions that require access to the operator's database ect.), and the rest of the SFC is deployed on the domain of other operators; the choice of the parts of the SFC that should be deployed externally may be motivated by additional factors like resource consumption cost, or the generated latency.

Post-Deployment In case the user request load increases, and the initially deployed SFC does not dispose of sufficient resources in order to respond to those requests while satisfying the predefined SLAs, the operator typically identifies bottelneck-inducing VNFs, then performs elasticity

operations by either allocating additional resources to those VNFs, or deploying additional instances for load balancing; however, there might be cases where the operator does not dispose of the required resources on its infrastructure. A solution would then be to scale-out those VNFs by deploying new instances on other operator's domains, and perform load balancing; or deploy new instances with more resources and re-route the traffic to these new VNFs.

1.1.2 Multi-Domain SFC Challenges

The multi-domain context introduces multiple challenges, that are mainly due to the autonomous management and orchestration of each domain, and the level of trust that is established with external entities. In this context, additional constraints apply as a result of the limited visibility and control on each local domain, and heterogeneous technologies and protocols. In the following, we describe some of the main challenges related to multi-domain SFC:

- For security reasons, the domains tend to hide key information on their infrastructure. They would disclose the amount of resources that they make available for the SFC, the price per unit, and information on their border gateways in order to reach the domain. However, the internal topology information is kept hidden. This makes the SFC placement process more difficult due to the lack of visibility. Indeed, the placement optimization process requires information on the deployment infrastructure to find the best solution. Therefore, new placement schemes and algorithms are required to perform the multi-domain SFC mapping with a limited visibility on the infrastructure.

- Similarly, the end-to-end deployment and management of the SFCs, as well as the orchestration architectures need adaptations in order to cope with the limited visibility and control that each domain allows on their infrastructure. New protocols are required for the deployment and configuration of multi-domain SFCs, in order to ensure the end-to-end inter-domain packet forwarding.

- Due to the independent management of the domains, different traffic routing and encapsulation methods might be employed by each domain, which requires the deployment and configuration of interfacing entities to ensure compatibility regardless of the implemented protocols and encapsulations of the domains.

1.2 Thesis Statement and Contributions

Based on the aforementioned challenges, an orchestration framework is needed in order to ensure the placement, deployment, and life-cycle management of multi-domain SFCs. In this thesis, we propose different solutions to enable Service Function Chaining over multiple domains:

- First, we tackle the issue of placing multi-domain SFCs with limited visibility on the infrastructure. We propose a hierarchical framework, that allows a logically centralized coordinator to collect information disclosed by each domain, and create an abstracted view in order to compute an initial placement of the SFC, then partition the SFC accordingly, and send each sub-SFC to the designated domains. In case the local placement of a sub-SFC fails, a backtracking mechanism is triggered in order to compute a new solution while ruling out the last combination.

- Second, we propose multiple methods and algorithms to compute the placement of SFCs. We formulate an Integer Linear Programming (ILP) model that minimizes the cost and end-to-end latency, while satisfying the service's constraints. Then, in order to better accommodate the user preferences, we extend the model using Physical Programming, and SLA classes that reflect the 5G use cases and their requirements in terms of latency and bandwidth per user. Both models are implemented using exact solutions, as well as heuristic algorithms for scalability.

- Finally, we devise on an ETSI-compliant multi-domain SFC deployment and orchestration framework, that includes an interfacing component in order to translate the SFC encapsulations at the ingress and egress of each domain, and ensure compatibility. We also detail the workflow for multi-domain SFC deployment that allows the configuration of the different architectural components to perform the end-to-end packet forwarding.

Together, the contributions of this thesis allow the elaboration of a multi-domain SFC deployment orchestration architecture that manages the pre-deployment and deployment phases of the SFC life-cycle.

1.3 Thesis Structure

This thesis is structured as follows: Chapter 2 provides an overview of the state of the art of SFC and its enabling technologies, and details the challenges related to multi-domain SFCs. Chapter 3 and Chapter 4 are dedicated to the optimal placement of multi-domain SFCs with limited visibility

on the infrastructure. Chapter 5 tackles the deployment phase, and proposes an orchestration architecture for multi-domain SFC. Finally, we conclude the document in Chapter 6 with an analysis of the thesis contributions, as well as an overview of the remaining open issues.

Chapter 2 - State of The Art: In this chapter, we introduce the main concepts and principles related to SFC, its architecture and enabling technologies, and the existing contributions in each phase of the SFC life-cycle. Then, we detail the challenges related to multi-domain SFC, as well as the existing works, and their shortcomings.

Chapter 3 - SFC Placement - Weighted Sum Approach: In this chapter, we propose a hierarchical framework for SFC placement with limited visibility. A centralized Multi-Domain Orchestrator creates an abstracted view of the network using the information provided by each domain, then performs an initial placement of the SFC before partitioning the request, and sending the sub-SFCs to the local domains accordingly. We then formulate an ILP model for the optimization problem, aiming to minimize the overall cost and end-to-end latency using the weighted sum approach. Finally, for scalability, we develop an efficient memetic heuristic algorithm to solve this problem. The work related to this chapter is:

- ***Accepted*** - "A Multi-Objective SFC Placement Scheme over Multiple Domains", *IEEE International Conference on Communications (ICC) 2019*, Shanghai, China, N.Toumi, D.Meddour, A.Ksentini.

Chapter 4 - SFC Placement - Physical Programming: In this chapter, the ILP formulation is improved to better express the user's preferences using Physical Programming, which allows the formulation of the preferences in terms of meaningful interval values. Three Physical Programming methods are used: Linear, Non Linear, and Global. In order to support the 5G use cases and their requirements, we leverage on SDO documents in order to define several SLA classes and their requirements in terms of latency and bandwidth per user. Finally, we develop a Branch and Bound exact algorithm to solve this formulation, as well as a heuristic algorithm in order to perform the multi-domain SFC placement, by minimizing cost and end-to-end latency, while maximizing the allocated bandwidth per user. The work related to this chapter is:

- ***Published*** - "On Using Physical Programming for Multi-Domain SFC Placement with Limited Visibility", *IEEE Transactions on Cloud Computing*, N.Toumi, O.Bernier, D.Meddour, A.Ksentini, doi: 10.1109/TCC.2020.3046997.

Chapter 5 - SFC Deployment and Orchestration: In this chapter, we propose an ETSI-compliant orchestration framework in order to deploy multi-domain SFCs while taking into account the difference in encapsulations implemented by each domain. We extend the Or-Or reference point as well as the Internet Engineering Task Force (IETF)'s Internal Boundary Node [27]. We then detail the mechanisms for the end-to-end SFC deployment and deletion, as well as the information that needs to be exchanged between the architecture components. Once the SFC has been deployed, we describe the end-to-end packet forwarding mechanism. Finally, in order to validate our architecture, we implement a Proof of Concept platform for multi-domain SFC deployment, where different encapsulation types are employed. The works related to this chapter are:

- ***Presented*** - "Towards Cross-Domain Service Function Chain Orchestration", *IEEE Global Communications Conference (Globecom) 2020*, Taipei, Taiwan, N.Toumi, O.Bernier, D.Meddour, A.Ksentini.
- ***Accepted***- "On Cross-Domain Service Function Chain Orchestration: An Architectural Framework", *Elsevier Computer Networks Journal*, N.Toumi, O.Bernier, D.Meddour, A.Ksentini.

Chapter 2

State of The Art

2.1 Introduction

As stated in the previous chapter, new technologies were developed to accommodate the new 5G use cases and applications, and satisfy their requirements. In the core part of the network, two main enablers have emerged: Software Defined Networking, and Network Function Virtualization. Together, these technologies form the basis for the deployment of 5G services, and enable other paradigms such as Network Slicing, and Service Function Chaining.

2.1.1 Software Defined Networking

SDN is a networking paradigm proposed by the Open Networking Foundation (ONF). It introduces a layered networking architecture where the control plane is separated from the data plane. In the control plane, a logically centralized controller, with global knowledge on the network, elaborates forwarding policies that implement the desired network behavior, formulated by network applications through the northbound interface using Application Programming Interfaces (APIs). The forwarding plane is composed of the interconnected forwarding devices that perform actions on packets according to the instructions of the control plane, communicated through a southbound interface using protocols such as OpenFlow [10]. This approach has multiple advantages: Since the network intelligence is centralized in the control plane, the data plane devices only apply the forwarding rules, which means that less intelligence is needed in the networking equipment, therefore network deployment costs can be reduced. The centralized control plane also makes the network programmable and allows more flexibility. Indeed, the global network behavior can be changed dynamically by updating the forwarding rules in the data plane devices.

2.1.2 Network Function Virtualization

NFV is a virtualization technology that decouples network functions from the underlying hardware equipment. Instead of using dedicated proprietary appliances, Virtual Network Function VNF instances can be deployed on commodity servers, thus reducing CAPEX and OPEX costs [11]. By virtualizing the network functions, their deployment can be performed in a more efficient manner. Indeed, network services can be deployed and migrated faster, and scaling benefits from added flexibility as resources can be dynamically provisioned to the virtual instances [12]. Based on this approach, Network Services (NS) are decomposed into sets of VNFs. In order to deliver these services, the VNFs ought to be interconnected using Connection Points (CPs), Virtual Links (VLs), and VNF Forwarding Graphs (VNFFGs) that describe the interconnection topology between VNFs. Furthermore, traffic rules need to be implemented in order to forward packets to each VNF of the network service in an ordered manner.
This process is referred to as Service Function Chaining. In this chapter, we provide an overview on the state of the art of SFC, and the previous contributions in the literature, then, we introduce the multi-domain context and its challenges.

2.2 Service Function Chaining

A Service Function Chain is an abstraction of a service that defines its different functional components (Service Functions), as well as the order (strict or flexible) in which the packets need to be processed by each component (Service Function Path) in order to deliver an end-to-end network service. The RFC7665 [28] defines a general architecture for the SFC data plane, which is based on the following principles:

- Topological independence: SFC deployment doesn't require any changes to the underlay topology.
- Plane separation between the SFP establishment (control plane) and the packet handling operations (data plane).
- Classification: Traffic that matches the classification rules for a given SFC is forwarded according to the specific SFP of that SFC. Classification rules can have different levels of granularity: it can use specific TCP/IP fields, parts of the packet's payload, or rely on higher-level inspections.
- Shared Meta-data: Meta-data/context data can be shared between different SFC components. An example of shared Meta-data is user/subscriber information.

- Service definition independence: The SFC architecture doesn't depend on the details of its service functions.
- Service function chain independence: Any operation on an SFC (creation, modification, deletion) doesn't impact other SFCs. This also applies to SFPs.
- Heterogeneous control/policy points: Independent mechanisms can be used to define local classification criteria and traffic steering policies.

The Service Function Chaining deployment architecture can be modeled in three layers:

- **Data plane:** Is responsible for traffic steering and treatment.
- **Control Plane:** Enforces the forwarding rules on the network components.
- **Orchestration and Management plane:** Determines the traffic steering rules, the placement of service functions, and manages their life-cycle events.

2.2.1 Data Plane of SFCs

2.1.1.1 Functional components

The general architecture for Service Function Chaining as described in the RFC 7665 [28] is illustrated in Figure 2.1, and is composed of the following core components:

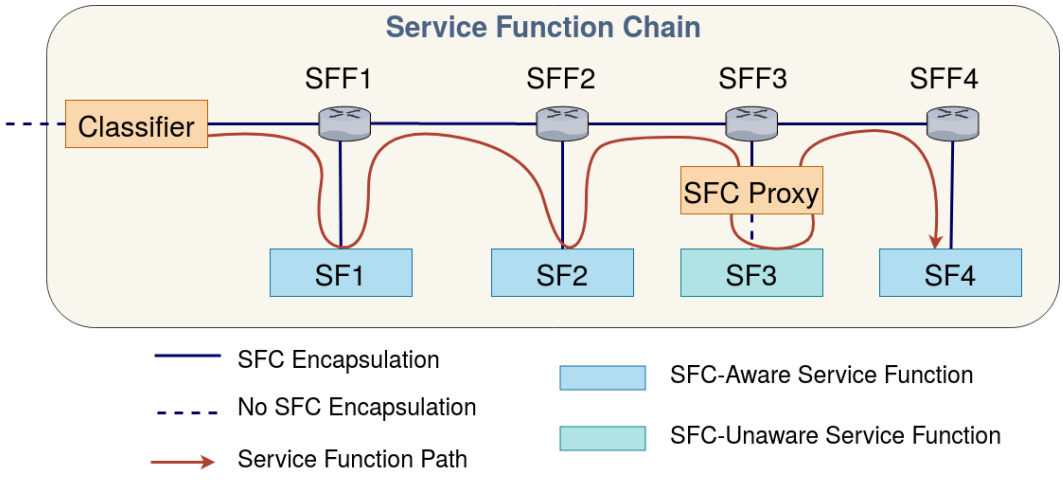


Figure 2.1: SFC Architecture in RFC 7665

- **Classifiers:** Each packet entering the network goes through a classifier that determines the SFC that a packet belongs to, and selects/creates the appropriate SFP. As mentioned earlier, the classification is done by matching pre-defined classification rules. Once the SFC and SFP are determined, the packet is encapsulated and forwarded to the first Service Function in the SFP.
- **Service Function Forwarders:** Are responsible for forwarding the received packets to the following SFF, or Service Function in the SFP, according to the information contained in the SFC encapsulation of the packet. In case the SFP allows multiple choices in forwarding, a SFF must ensure that all the packets are treated the same way.
- **Service Functions:** Are logical components responsible for performing specific treatment of ingress traffic (e.g. firewalls, DPI, Network Address Translation (NAT), Load Balancers), after processing the packets, the SFC encapsulation is updated in order to steer the traffic to the next hop in the SFP, and the packets are sent back to the SFF. Note that a Service Function can be assigned to more than one SFC. Some Service Functions are able to reclassify packets; for example, if a DPI detects that a certain traffic is malicious, it reclassifies it accordingly so that the packets are dropped or redirected to a firewall.
- **SFC Proxys:** In case some Service Functions are "SFC-unaware", which means that they don't support the SFC encapsulation; packets being forwarded to these service functions need to pass through proxies that remove the SFC encapsulation, then add it again to the classified egress packets with updated information and forward it to the next hop in the SFP.

2.1.1.2 Traffic Steering

Data plane forwarding components forward SFC traffic according to traffic steering policies. Traffic steering in SFC can be performed using two key approaches [29, 30]:

Flow-based traffic steering: This approach preserves the original packets, and the forwarders steer packets after identifying the SFC they belong to. A few proposed solutions rely on OpenFlow classification [29, 31–34] for SFC identification; another solution relies on Deep Packet Inspection [35] at the switch level for a more fine-grained classification. However, this latter solution introduces significant delays, as the classification needs to be performed by the forwarding components at each hop. Furthermore, it supposes that the forwarding elements dispose of sufficient processing capabilities in order to perform complex packet inspection and classification.

Packet-based traffic steering: This approach assumes that the packet includes header fields that can be used in order to determine the next hop in the SFC. This identification information can be added using tag-based methods, where header fields of existing encapsulation methods are used for traffic steering; and header-based methods, where new encapsulation headers are defined for SFCs:

- **Tag-based methods:** The SFP is inserted in the form of MAC [36], VLan [37] or Multi Protocol Label Switching (MPLS) [38] tags. One commonly used tag-based method is Source Routing [39] or Segment Routing, where the list of SFs is encoded as a stack of tags or labels such as MPLS [40]. The disadvantage of these methods lies in the encapsulation size that increase with SFC length, which could lead to packet fragmentation issues due to small MTU (Maximum transmission unit) values .

- **Header-based methods:** Another method consists in adding specific headers that allow the identification of the SFC. One widely implemented solution is the Network Service Header (NSH) encapsulation [41], a similar encapsulation is the Service Chain Header (SCH) [42]. Other proposed encapsulation headers leverage on IPv6 and its extension header in order to encode path information such as in NSH or SCH [43], or by encoding the list of IPv6 addresses of the SFC elements that the packets need to be steered through [44]. However, the use of IPv6 encapsulations increases the total packet size due to the IPv6 address and header size.

In the following, two examples of packet-based encapsulation methods are detailed:

Network Service Headers: The Network Service Header [41] encapsulation is illustrated in Figure 2.2. The SFC Identifier is encoded into the 3-byte Service Path Identifier (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 encapsulation also allows the inclusion of metadata, with two metadata types: A fixed 16-byte metadata field for MDType 0x1, and a variable metadata field that includes its own header for MDType 0x2 that includes Metadata Class, type, and length information. This header is added upon the classification of packets at the ingress of the SFC, and each time the packet passes through a Service Function along the chain, the SI is decremented. Note that the NSH header can encapsulate IPv4,IPv6, Ethernet, MPLS packets, but also NSH packets (Next Protocol: 0x1,0x2,0x3,0x5,0x4 respectively), which means that a multi-level NSH can be implemented.

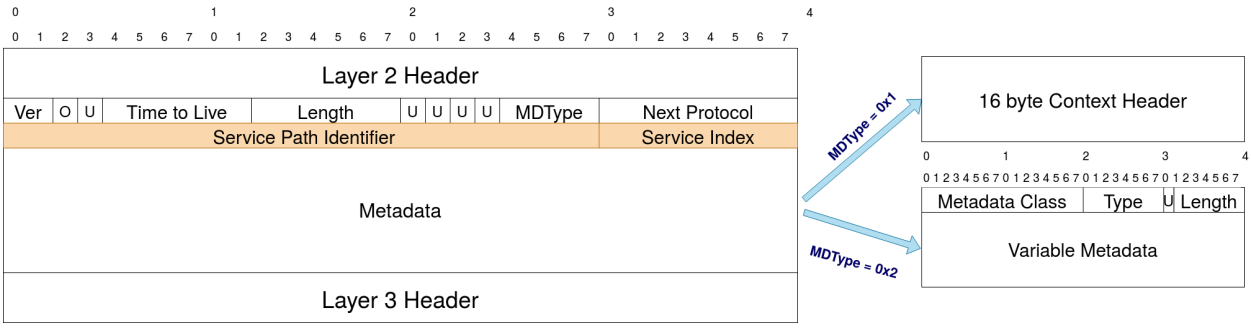


Figure 2.2: Network Service Header Encapsulation

Segment Routing: In Segment Routing [39, 40, 45, 46] 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, as well as the network convergence time and configuration messages. However, this method lacks scalability as it significantly increases packet size, especially for long SFCs; it also lacks flexibility as the path is statically specified at the source node. In [38], the authors leverage on MPLS tags in order to implement an SDN-based source routing solution for SFC. The authors address the scalability issue by dividing chains into *pathlets* and allowing intermediate nodes to insert source routing information. The authors in [44] also propose a solution for Service Chaining that leverages on Segment Routing, and the solution provides support for Segment Routing unaware VNFs.

Traffic Steering Rules: In order to perform traffic steering, the SDN paradigm can be leveraged on, where Openflow rules are pushed to the SDN enabled switches. Depending on the encapsulation, the headers fields are used by the SFFs in order to determine the next hop. Figure 2.3 illustrates the forwarding rules, as well as the encapsulation headers at each hop of the SFC, for two different SFCs, and both the NSH and Segment Routing encapsulations. For the first SFC, the packets flow through SF1, SF3, and SF4, and the second SFC is composed of SF2, SF3, and SF4. The traffic steering rules for NSH allow the Service Function Forwarders to determine the next interface to forward the packets to based on the SPI and SI values. At each hop, the SFs decrement the SI value before sending the packets back to the SFF. In contrast, for the Segment Routing implementation, the list of hops is directly encoded into the packet headers, which means that the forwarding rules are fewer and less complex, and the next hop computation is shorter. At each SF, after the processing of packets, the top label in the list of hops is popped, so that the next hop becomes at the top of that list, and the packet is sent to the connected SFF. However, the encapsulation size at the beginning of the SFC might be bigger, depending on SFC length.

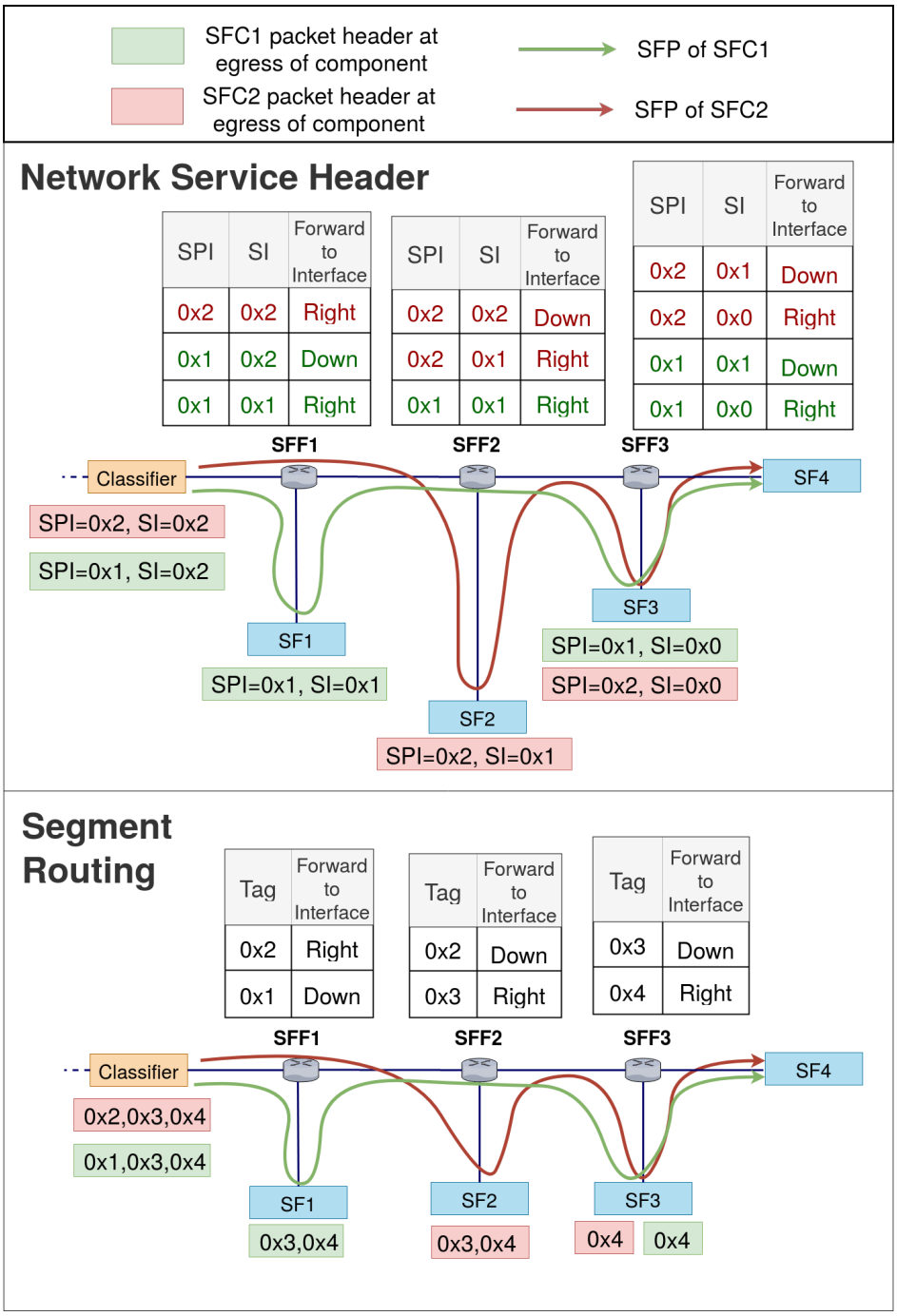


Figure 2.3: SFC Forwarding Rules

2.2.2 Control Plane of SFCs

The implementation of header-based methods such as NSH requires a control plane that ensures routing consistency within the network by constructing the map of SPIs, as well as communicating SPI values and forwarding policies to Classifiers, Proxies, and SFFs. This function can be performed by a SDN controller [47]. As shown in Figure 2.4, the controller communicates the classification and forwarding rules to the SFC components through interfaces that have been specified in an IETF draft [48] as follows:

- **C1 Interface:** The C1 interface between the control plane and the classifier is used to manage the classification rules.
- **C2 Interface:** The C2 interface between the control plane and the SFF is used in order to push the forwarding rules, and collect state information. In the SDN paradigm, this interface represents the southbound interface between the SDN controller, and the OpenFlow switches.
- **C3 Interface:** The C3 interface between the control plane and SFC-aware service functions is used to collect information on the processing of the packets.
- **C4 Interface:** The C4 interface between the control plane and the SFC proxies is also used to push forwarding rules and retrieve state information.

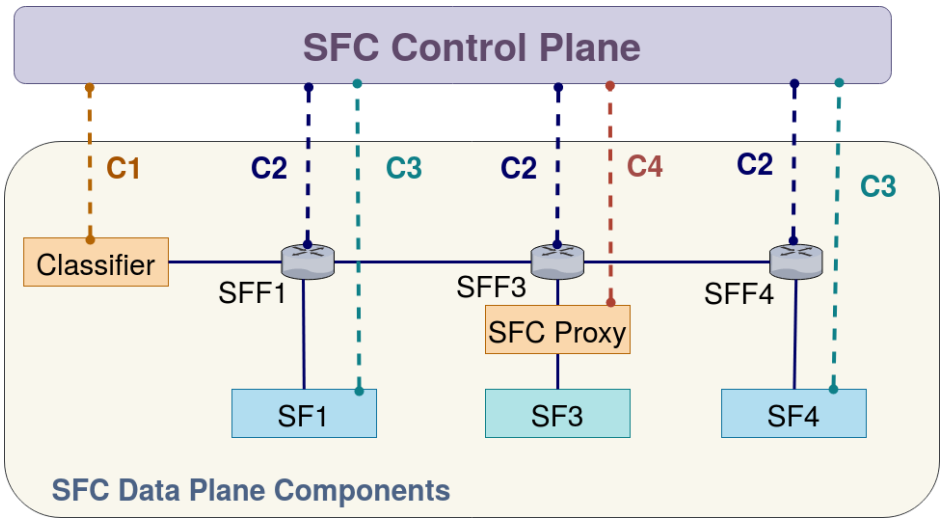


Figure 2.4: SFC Control Plane

Another draft [49] proposes an architecture for SFC implementation in which a SFP Manager is deployed as an intermediate between the management plane orchestrator and the SDN controller. This component computes Service Function Paths that interconnect the VNFs of the SFC determined by the orchestrator, then forwards the configuration to the SDN controller. Davoli *et al.* [50] propose an OpenFlow-based NSH control plane implementation where each service plane entity is assumed to support the OpenFlow protocol. The interconnection is done through tunneling technologies. Each SPI/SI pair is then associated with an OpenFlow switch port. In [51] the SPI field is re-purposed in order to identify the service function type only. The network would have to dynamically select the best instance according to the service function type, as well as the context data; this solution reduces the number of SPI values that need to be managed, but doesn't ensure end-to-end path visibility.

2.2.3 Management Plane

2.1.3.1 ETSI-MANO Architecture

The European Telecommunications Standards Institute proposed a framework for the Management and Orchestration (MANO) of Virtual Network Functions [12, 52, 53]. The proposed architecture is comprised of multiple functional blocks that can interact with other external functional blocks via reference points as shown in Figure 2.5:

NFV-MANO functional blocks:

Virtualized Infrastructure Manager (VIM): The Virtualized Infrastructure Manager is responsible for controlling and managing the NFV Infrastructure (NFVI) compute, storage and network resources (allocation/upgrade/release/reclamation), and managing the association of the virtualized resources to the physical compute, storage, networking resources. The VIM is also responsible for managing VNF Forwarding Graphs (create, query, update, delete), e.g. by creating and maintaining Virtual Links, virtual networks, sub-nets, and ports, as well as the management of security group policies to ensure network/traffic access control.

NFV Orchestrator (NFVO): The NFVO has two main responsibilities: the life-cycle management of Network Services (NS) (instantiation, update, query, scaling, collecting performance measurement results, event collection and correlation, termination) as well as VNF Managers, and the orchestration of NFVI resources across multiple VIMs, and governance of VNF instances sharing resources of the NFV infrastructure.

VNF Manager (VNFM): The VNF Manager is responsible for the life-cycle management of VNF instances. Each VNF instance is assumed to have an associated VNFM. VNFMs may be assigned the management of one or more VNF instances of different types.

Element Management System (EMS): The Element Management System is responsible for FCAPS (Fault, Configuration, Accounting, Performance, Security) management functionality for a VNF.

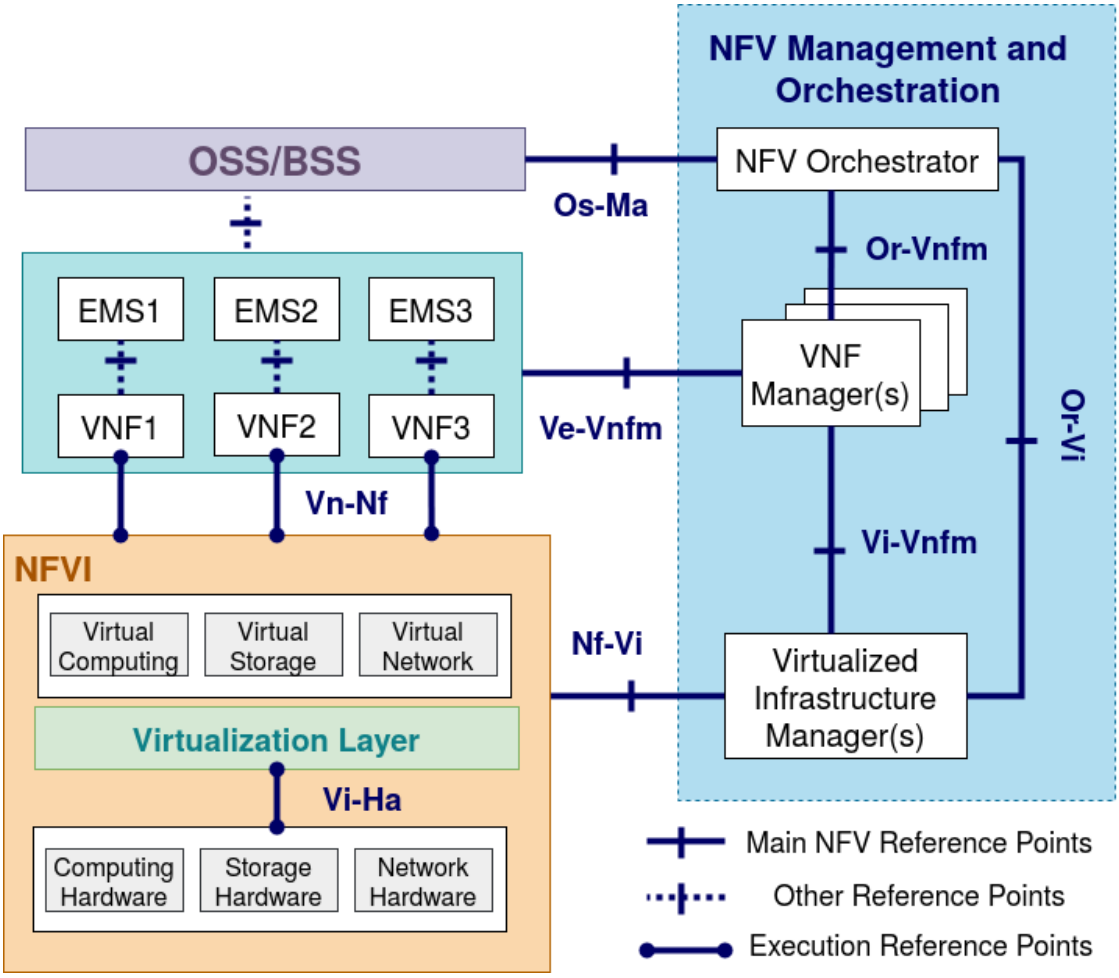


Figure 2.5: ETSI-MANO Architecture

NFV-MANO information elements: An example of a Network Service and the main information elements describing its components and their relationships and dependencies, as defined in the ETSI specification, is given in Figure 2.6. The Network Service is composed of 3 VNFs, that are interconnected through 2 Virtual Links and 4 Connection Points. The Network Service Descriptor (NSD) that describes it contains references to the respective VNF Descriptors, VL Descriptors, VNFFG Descriptors, and Connection Point information. The needed information elements are detailed below.

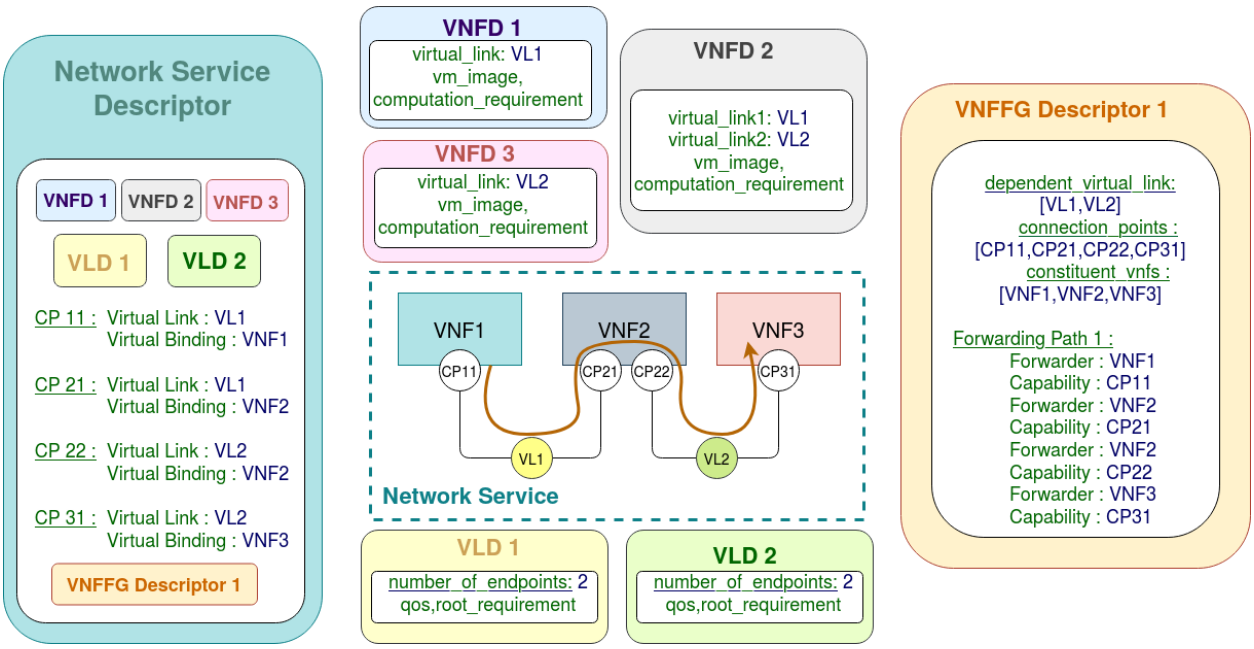


Figure 2.6: ETSI-MANO Network Service Descriptor and its Information Elements

Network Service Descriptor: To deploy a given service, a Network Service Descriptor is created in order to describe the service components such as VNFs and Virtual Links and their resource requirements, as well as the required connectivity and dependencies between these components. It is used by the NFV Orchestrator to instantiate an end-to-end Network Service. It consists of static information elements such as the ID, vendor, and version, references to the VNFDs, VLs, and VNFFGs that are detailed below. The NSD also describes the deployment flavors (i.e compute, memory, and storage capacity) of Network Services.

VNF Descriptor: A VNF Descriptor (VNFD) is a deployment template that describes a VNF in terms of deployment and operational behavior requirements. Some of the base information elements are the VM image that is used to deploy the VNF, its resource requirements, its connection points, as well as functional scripts for specific life-cycle events.

Virtual Link Descriptor: A VLD provides the connectivity parameters, and QoS and bandwidth requirements for the Virtual Links in the Network Service.

VNF Forwarding Graph Descriptor: The VNFFG is an information element that references the different VNFs, PNFs, as well as the virtual Links and Connection Points included in the Network Forwarding Paths of a given Network Service. Indeed, in order to provide a given Network Service, packet flows can follow more than one Network Forwarding Path. The VNFFG Descriptor can optionally specify the applicable forwarding rules over the NS topology; and define an ordered sequence of connection points to be traversed by the NS packet flow.

SDN Control Plane Integration As mentioned earlier, the SDN paradigm consists of separating between the network's control plane and data plane. The forwarding decisions are taken by a logically centralized controller that communicates those instructions to the data plane. The integration of the SDN Control Plane (SDN controller) into the ETSI NFV MANO framework has been explored in a specification [54] that enumerates the different possibilities, where the SDN controller is integrated into the VIM, the NFVI, the OSS/BSS, or deployed as a VNF. Depending on the integration option, the communication between the NFVO and the SDN controller would either use pre-existing interfaces, or require the creation of new interfaces.

2.1.3.2 SFC Support

The different functional components and information elements of the ETSI MANO framework can be mapped to the IETF notations as follows [55]:

ETSI-NFV	IETF
Virtualized Network Function (VNF)	Service Function (SF)
Connection Point (CP)	Service Function Forwarder (SFF)
VNF Forwarding Graph (VNFFG)	Service Function Chain (SFC)
Network Forwarding Path (NFP)	Service Function Path (SFP)

Table 2.1: Mapping between ETSI and IETF notations

Several works presented orchestration frameworks that support Service Function Chaining. In [56], a resilient SFC orchestrator based on the NFV-MANO framework is proposed. The architecture is extended in order to integrate an SFC Orchestrator that supports Service Chains. This additional component is responsible for mapping the SFP with the VNF Forwarding Graph, and creating policies that will be enforced in the SFFs through the SDN controller. In a subsequent work [57], a Scaling module is also added to the architecture. This module triggers scaling actions, and ensures SFP adaptation, as well as QoS-aware Load Balancing. The authors in [58] also present an orchestration framework for anomaly detection in SFC based on ETSI's NFV-MANO reference architecture, where a module is responsible for monitoring and maintaining SFC integrity. In [59], an orchestration reference architecture supporting SFC across multiple domains is presented. The architecture includes a Northbound interface for the VIM that allows the orchestrator to launch SFC creation by communicating its description; SFCs are described following a JSON template that specifies the source, destination, QoS requirements, as well as the list of VNFs with their characteristics.

In [60], the authors propose an orchestration framework for SFC that is ETSI-MANO compliant and integrates a user request parser, a request manager, placement and monitoring modules, as well as a simulator that supports the integration of placement algorithms. Wang *et al.* [61] propose an SFC orchestration system for IoT deployments that aims to reduce resource idleness while satisfying performance requirements. Medhat *et al.* [62] detail X-FORCE, an ETSI NFV-MANO compliant SFC orchestration framework that enables SFC life-cycle management and orchestration by integrating an SFC orchestrator to the reference architecture.

2.3 SFC Life-cycle

The life-cycle of a Service Function Chain can be modeled as illustrated in Figure 2.7. It is composed of two main phases: The pre-deployment phase, which concerns resource allocation, is the phase where the optimal chain composition and placement are determined in order to provide the requested service to the client, while fulfilling its requirements. The life-cycle management phase encompasses the deployment operation, the runtime operations such as monitoring and fault management, as well as the termination of the SFC. The whole process is coordinated by a management/orchestration entity.

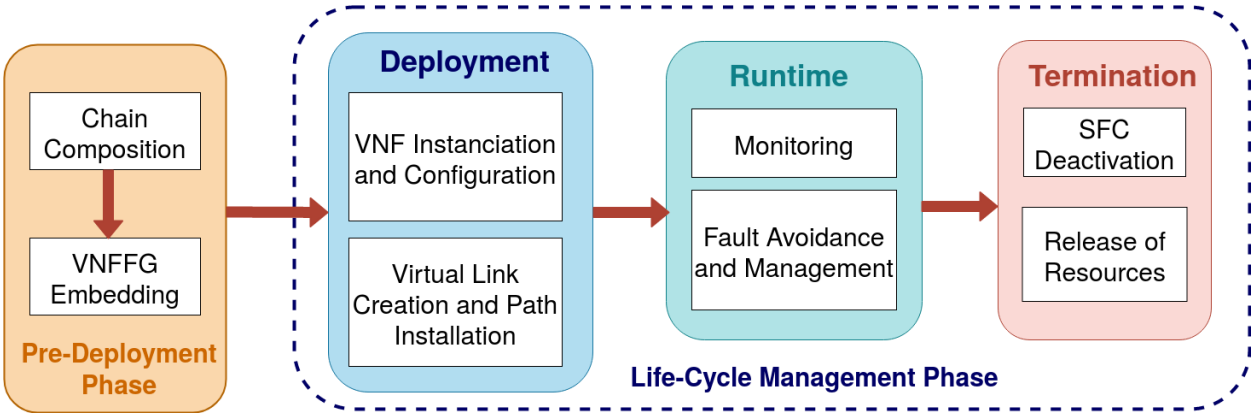


Figure 2.7: Life-cycle on a Service Function Chain

2.3.1 Pre-Deployment Phase

When a client emits a service request with a given SLA, the service provider must determine the optimal resource allocation and placement scheme. This challenge is known as the NFV-RA problem [63], which includes the VNF Chain Composition, and VNFFG embedding operations.

2.2.1.1 VNF Chain Composition

In this stage, the goal is to compose an optimal chain that answers the customer's request. The chain composition process defines the types of VNFs that constitute a given SFC, as well as their order (therefore producing the VNFFG), and the required amount of resources. A few works [64–69] studied the possibility of automatic SFC composition, while taking into account the dependencies between the VNFs. In [64, 65] the authors also explore the possibility of parallelizing VNFs, and the authors of [67] use behavior modeling and real-time verification and modification in order to ensure constraint compliance even after deployment.

2.2.1.2 VNFFG Embedding

Resource Assignment: It is essential to determine the amount of needed resources for each VNF in order to properly deliver the required service. Indeed, one under-performing VNF would cause a bottleneck that affects the whole chain, even if the other VNFs are assigned sufficient resources. Furthermore, resource allocation constraints may require redundancy in order to ensure resiliency [70]. In [71], bottlenecks in networks are identified using two approaches: power-off

based where the network state is examined when each node is powered-off to identify the most important nodes, and utilization-based, where the resource consumption of each node is studied in order to assign the proper amount of resources. More resources are then supplemented to the most sensitive nodes in the network. Similarly, in [72–74], key functions of SFCs are identified, and back-up instances of these functions are deployed. Alternatively, Wen *et al.* [75] present a VNF consolidation solution that aims at reducing resource over-provisioning that incurs due to fluctuations in service requests, while minimizing reconfiguration times and satisfying end-to-end performance requirements. And in [76], the minimum number of on-site backup Virtual Machines is provided in order to reach the required end-to-end availability of the SFC.

Resource Placement: This stage considers the mapping of the VNFs and Virtual Links to physical resources considering one or more optimization objectives. The VNF Placement problem has been proven to be NP-Hard [77] [78]. Many works have treated this problem using several exact or heuristic approaches; exact methods like Integer Linear Programming are used in order to formulate the problem, but due to their lack of scalability (due to the combinatorial explosion of the state space), heuristic algorithms are applied in order to obtain approximate solutions in polynomial times. The solutions take into account different functional and non-functional constraints in order to optimize the VNF placement. Other contributions such as in [79, 80], also explored multi-domain Virtual Network Embedding (VNE), which is the embedding of undirected graphs of connected VNFs. However, VNF placement in an SFC context adds a few more constraints; indeed, the SFC order and dependencies need to be considered to reduce the network delays, and optimize the SFP [55, 70, 81]. Furthermore, Service Function placement and routing might be subject to affinity and anti-affinity constraints for different reasons such as performance, availability, or legislation and privacy concerns [82], therefore, the optimization model needs to be adapted to the SFC framework [83]. In addition, if SFCs are expected to enable the deployment of 5G use cases, their placement scheme should support the different QoS requirements of these 5G use cases such as latency, and bandwidth.

A large set of works have addressed the SFC placement issue, with different performance optimization parameters such as cost [55, 78, 84–86], energy [87, 88], latency [89, 90], resource consumption [81, 91–94], security [95], or resiliency [20, 56, 96–98].

Some works also considered traffic fluctuations in their placement schemes. The authors of [99] formulate a model for SFC deployment and readjustment to address user mobility or service request fluctuations. Li *et al.* [100] consider the workload correlation between different service requests in order to avoid mapping requests that have coinciding peaks jointly. Huang *et al.* [101] propose a

solution for SFC deployment that also takes into account the correlations between VNFs that can result from coordination between them or traffic changes. The authors in [102] propose a randomized approximation algorithm that performs SFC embedding with the objective of maximizing the amount of flows processed by network functions.

Joint Composition and Placement: A few works also combined two stages of the NFV Resource Allocation problem [77, 103–113]. The goal being to take advantage of the order flexibility for some VNFs, and the possibility of decomposing VNFs into elementary functions, in order to group VNFs in a way that minimizes resource fragmentation and consumption.

2.3.2 Deployment

As shown in Section 2.2.3, SFC deployment can be performed by ETSI MANO compliant orchestration frameworks, by expressing the SFC elements and their requirements and connectivity in terms of information elements of an NSD. The SFC would then be deployed as an NS on the Virtual Infrastructure, by leveraging on the standard NS deployment workflows.

Further, in order to perform SFC packet forwarding, the SDN paradigm can be used as detailed in Section 2.2.1 by pushing the forwarding rules to OpenFlow enabled switches.

2.3.3 Runtime

During runtime, and in order to serve the ingress user service requests in the most effective way while respecting the client's SLAs, the SFC processing schedule needs to be optimized; on the other hand, service requests fluctuations as well as the advent of anomalies may require to trigger different elasticity mechanisms and/or dynamic path adaptation.

Monitoring and Anomaly Detection

Multiple works proposed anomaly detection frameworks based on periodical analysis of the SFC state, to detect overload or failure [56, 58, 114]. The works in [115, 116] implement verification tools leveraging on the NSH encapsulation in order to verify traffic steering rules, and detect connectivity issues. In [117] the authors develop a predictive analysis framework for network troubleshooting and diagnosis. The analysis focuses on the forwarding behavior of VNFs.

Elasticity Mechanisms

In case an anomaly has been detected, it is necessary to take actions in order to avoid service disruption, or the degradation of the QoS and QoE. Furthermore, in case the traffic volume increases to a certain point, the service functions might need additional resources in order to deliver the service properly. Alternatively, resources should be released in case the traffic load decreases. Multiple elasticity operations can be considered [118, 119]:

- **Scale-Up/Down:** Allows the addition or removal of resources from a virtual instance.
- **Scale-In/Out:** Allows the addition or removal of a virtual instance of the same virtual function, with the implementation of load balancing between the instances to distribute the traffic to each instance.
- **Migration:** Migration is a process in which a virtual instance is moved from a physical node to another, with minimal service disruption, and while preserving its state.

In [120], the authors propose a model to determine whether it's best to migrate or re-instantiate a given VNF in order to minimize the downtime, and the authors of [121] present a migration policy that determines when and where VNF migrations should be performed. Other works [122–125] propose orchestration schemes that detect and respond to request fluctuations, as well as node and link failure using mitigation mechanisms such as scaling, migration, or congestion control. Nam *et al.* [126] investigate the performance effects of adding VNFs to an SFC, or changing the VNF sequence and propose an automated analysis approach that identifies the root cause of performance issues.

Dynamic Service Path Selection

The SFC path selection problem is different from the traditional routing problems; indeed, SFC paths can require a strict order of visited VNFs, and an SFC flow might need to visit the same VNF more than once in the case of non-linear SFCs. In [127], the authors argue on the necessity of a dynamic traffic steering mechanism for existing and arriving flows, that considers the traffic load as well as the computation load on the nodes. The works in [128–131] implement dynamic SFC routing schemes and flow prioritization in order to guarantee the required QoS levels. In [132], the authors development online and offline traffic routing algorithms that also implement load balancing. Similarly, the work in [133] presents a model for an optimal flow distribution across the network in order to minimize end-to-end flow latency and flow dropping rate. Yi *et al.* [134] propose a solution for path adaptation in case of scale-in and scale-out operations. Two methods

are implemented: a reactive scheme SFC-RS that doesn't change the original SFP, and a proactive scheme SFC-PS that optimizes the SFP

2.4 Multi-Domain Context: Definition and Challenges

As previously detailed, many research works explored Service Function Chaining under different aspects: composition, resource allocation, as well as placement and chaining, and proposed solutions in order to enable and support SFCs. However, most of these works assume that the SFC is deployed on a single domain and ignore the multi-domain scenario, which remains an open issue with many challenges that need to be tackled in order to provide end-to-end services.

The multi-domain scenario occurs when the SFC requires the deployment of its components on different domains. As previously stated, a domain is defined as an autonomous network infrastructure, with an orchestration entity that independently implements its own management decisions, and its own networking protocols. Therefore, the multi-domain context can be applied to multiple administrative entities, and also to separate divisions of the same entity. This scenario brings out multiple challenges, caused by the limited visibility and control on the infrastructure of the local domains, as well as the independent management of the domains. In consequence, during the life-cycle of an SFC, the multi-domain context needs to be taken into account from its placement during the VNFFG Embedding operation until its termination, and the placement, deployment, and runtime operations ought to be adapted in order to support multi-domain SFCs.

2.4.1 SFC Placement

As shown in Section 2.3.1, multiple works tackled the SFC placement issue, with different optimization parameters. However, most of these works consider placement over a single domain, and assume that the orchestrator performing the placement disposes of full visibility and control of the underlying network infrastructure. As previously outlined, deploying SFCs on multiple administrative domains adds more constraints to the placement problem. Indeed, for security reasons, the cloud providers withhold details on their local infrastructure, which makes it difficult to determine the optimal end-to-end placement and chaining of services due to insufficient information.

Furthermore, in addition to the support for multi-domain architectures with limited visibility, multi-domain SFC placement solutions are required to jointly optimize multiple KPI metrics related to the use case of the SFC, in order to meet the QoS requirements of the service, while reducing deployment costs.

2.4.2 Deployment

Once the placement scheme of the multi-domain SFC has been determined, the original NSD needs to be partitioned into a set of sub-NSDs, and each domain orchestrator undertakes the task of deploying its assigned sub-SFC locally. However, the domain orchestrators should also be able to identify the SFC and its sub-SFC, as well as the next hop corresponding to the next sub-SFC, in order to properly configure their forwarding and classification components. This presumes that a communication protocol between orchestrators is in place to enable the sharing of the required information. The ETSI framework was extended in its third release [135] 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 and its sub-SFC, 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 end-to-end packet steering is ensured.

As for SFC packet forwarding, multiple traffic steering methods have been proposed as previously detailed in 2.2.1. But although these solutions enable SFC packet forwarding inside a single domain, they do not support multi-domain SFC packet forwarding. 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 which ensure the end-to-end service delivery, regardless of the underlying technologies. To this aim, they must perform inter-domain packet forwarding, encapsulation/decapsulation or tunneling of packets, as well as communication protocol negotiation.

The RFC 8459 [27] attempts to tackle the multi-domain issue by proposing a hierarchical multi-level network architecture for SFC, which allows the decomposition of a large network into a set of independent sub-domains as shown in Figure 2.8. 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 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. Furthermore, this approach assumes that the interconnection domain is SFC-aware, which might not always be the case.

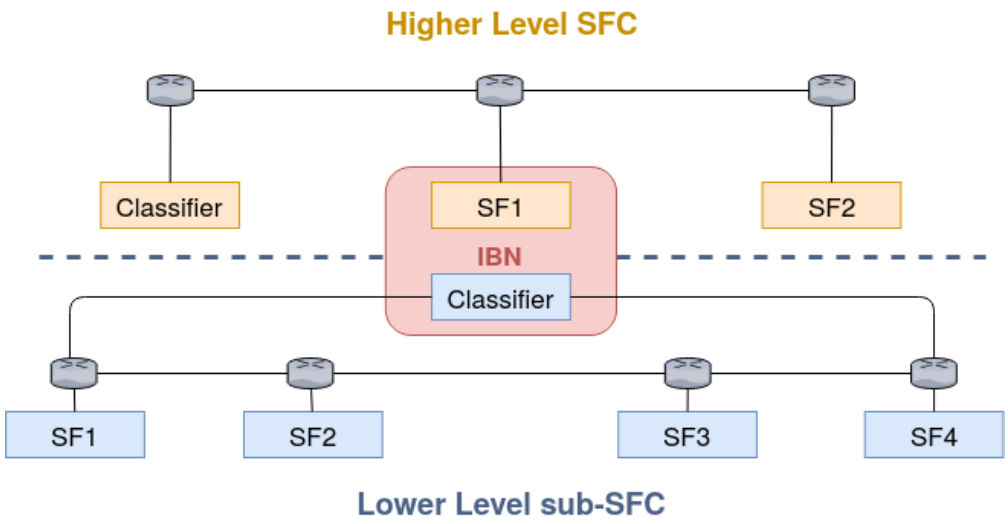


Figure 2.8: Hierarchical SFC Architecture in RFC 8459

Similarly, [136] proposes a horizontal approach for multi-domain NSH. SFCs that span multiple domains are divided into a set of single-domain segments. The identifier of the next domain's classifier is encapsulated in the metadata part of the NSH encapsulation (MDType 0x2) using the Metadata Class 0x20, and Type 0x1, as shown in Figure 2.9:

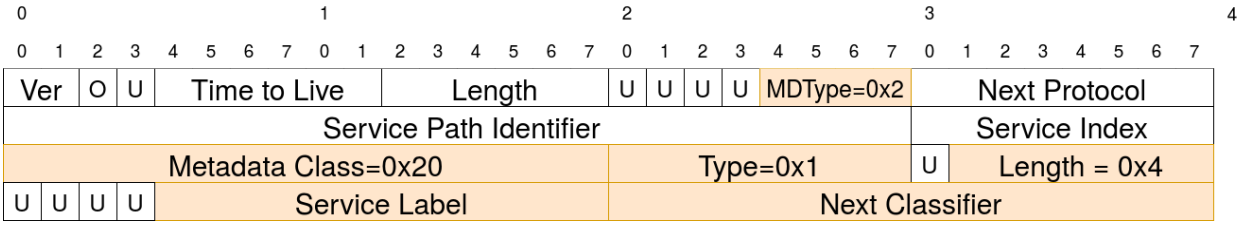


Figure 2.9: Multi-domain NSH encapsulation

The next classifier field contains the ID of the classifier in the next domain, and the service label is used by this classifier in order to properly identify the SFC. However, this architecture can only be applied to NSH-aware domains and inter-domain interconnections.

Based on all of the aforementioned challenges, an orchestration and deployment scheme is needed to enable multi-domain SFCs, with an information sharing mechanism in order to ensure end-to-end packet forwarding, as well as an encapsulation translation scheme in order to support heterogeneous encapsulation types.

2.5 Conclusion

In this chapter, we introduced the key concepts related to Service Function Chaining, as well as its enabling technologies. We also gave an overview of the literature, and identified the main challenges of SFC deployment over multiple domains for different stages of the SFC life-cycle.
In the next chapter, we propose a framework for multi-domain SFC placement, and implement an ILP model, as well as a heuristic in order to perform SFC placement while optimizing cost and latency using the weighted sum approach.

Chapter 3

SFC Placement - Weighted Sum Approach

3.1 Introduction

The deployment of an SFC requires the placement of its constituent VNFs, and path establishment in order to connect those VNFs, in a way to optimize the operator's objectives while satisfying its constraints. In the multi-domain context, the SFC is partitioned into sub-chains embedded in different domains, then chained together in order to ensure the end-to-end service. A few works have addressed the multi-domain SFC deployment issue with limited visibility on the network. Two main architectural approaches were proposed:

Distributed

This approach supposes that the infrastructure providers don't share any details on their network. Hence, 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. However, this approach falls short in terms of scalability, as communication and convergence time and cost are considerable. In [137], the authors propose a distributed non-cooperative congestion game model that performs SFC deployment without full knowledge of the user's information. Therefore, the orchestration task is distributed among Network Service Brokers that cover delimited portions of the network. The work in [138] details a policy-based, distributed, asynchronous election protocol based on hosting capabilities; the solution allows edge and core cloud providers to cooperatively instantiate wide-area chains. However, the proposed solution does not support more complex, non-linear SFCs, where packets pass through certain functions more than once, and its placement evaluation only considers CPU and bandwidth constraints. Zhang *et al.* [139] also propose a distributed vertex-centric algorithm that supports SFC flexibility: the request is relayed between orchestrators

in order to determine the optimal placement combination. The authors in [140] detail DistNSE, a distributed framework that performs SFC partitioning and placement using a privacy-preserving bidding mechanism where each Network Function Provider (NFP) competes for the s of the chain.

Centralized

In this approach, a broker/coordinator collects the information disclosed by different IaaS providers, and reconstitutes an abstract global view of the network. The centralized broker performs an initial placement using this abstract view, then partitions the request and relays the sub-requests to the local domains. However, this approach might lead to sub-optimal orchestration decisions due to the unsufficient information on the infrastructure state. Figueira *et al.* [141], and Guerzoni *et al.* [142] propose a hierarchical architecture for multi-domain SFC orchestration, where a centralized main orchestrator interfaces with lower-level domain orchestrators. Dietrich *et al.* [143] leverage on this architectural approach and detail a solution for SFC mapping across datacenters that are operated by multiple NFPs. The proposed solution allows NFPs to disclose minimal information about their infrastructure, and constructs an abstract view of the network topology. The placement is then performed in two stages: graph partitioning and sub-graph mapping; however, the solution doesn't take latency into account, which is a critical requirement for some upcoming 5G use cases (i.e. Ultra Reliable Low Latency services). Furthermore, the solution is obtained using Linear Programming optimization, and therefore lacks scalability due to exponential computation times, which makes it unsuitable for bigger instances of the problem. Similarly, Xu *et al.* [144] propose a multi-domain service chain partition and embedding scheme using a Hidden Markov Model and a Viterbi-based heuristic. However, the proposed heuristic solution only considers the end-to-end latency while discarding cost. In [145], the authors propose a multi-domain SFC orchestration scheme that takes into account the issues related to the lack of visibility over domains. They propose an algorithm that aims to minimize the delay, and improves the bandwidth cost of the obtained solution. However, this algorithm does not take into account the service and SLA types of the requests. Indeed, a Best Effort service, for example, would require the minimization of cost over latency.

In this chapter, we formulate the multi-domain SFC placement problem and elaborate multiple algorithms that perform placement with limited visibility, while minimizing the end-to-end delay and cost, and maximizing the bandwidth per user.

3.2 Proposed Framework

3.2.1 Architecture

In this section, we detail our proposed framework for multi-domain SFC deployment. As illustrated in Figure 3.1, the architecture is logically centralized. A multi-domain orchestrator acts as a broker and constructs an abstract view of the topology based on the information that each domain is disposed to reveal on their network. The orchestrator also interacts with the WAN domain operators in order to establish inter-domain communication for SFCs. Based on the work in [146], we suppose that the local domains disclose the following details on their infrastructure:

- The amount of computing capacity that is made available, with the average cost per unit for each resource type. Note that these amounts are determined through pre-established mutual agreements between the domain operators.
- The vertices of their WAN links, the available capacity, latency, and the cost per bandwidth unit.

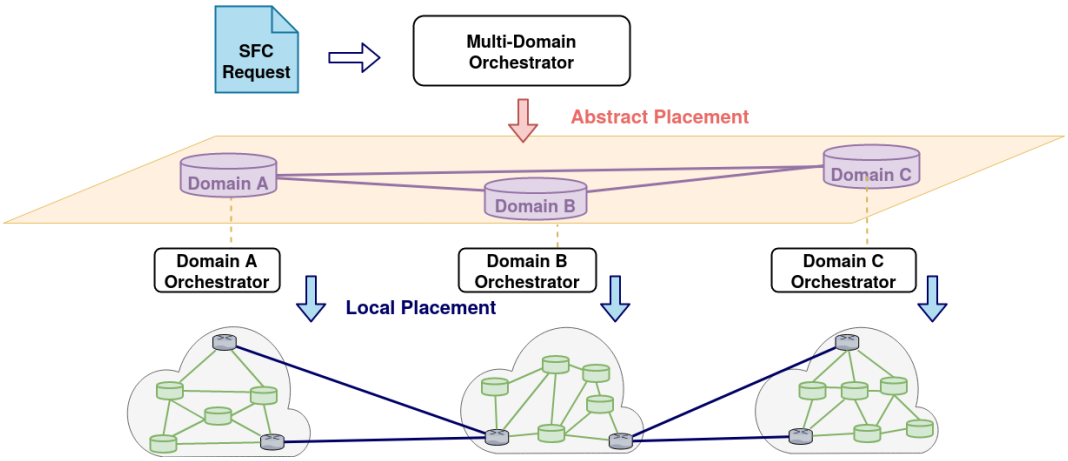


Figure 3.1: Multi-domain SFC Placement Framework

Note that the centralized approach means that all of the SFC deployment requests are sent to the Multi-Domain Orchestrator. However, this centralized architecture doesn't suffer from the problems related to scalability, as these requests are formulated by tenants who intend to deploy services that would accommodate multiple users. The number of service deployment requests from

the tenants is rather small compared to the service requests by the end users. Therefore, we can consider that the multi-domain orchestrator is able to handle the SFC request flows. Once an SFC is deployed, it is up to the tenant to provide the service users with an interface that allows them to have access to the service, without going through the centralized coordinator. The latter doesn't process the individual user requests for the service.

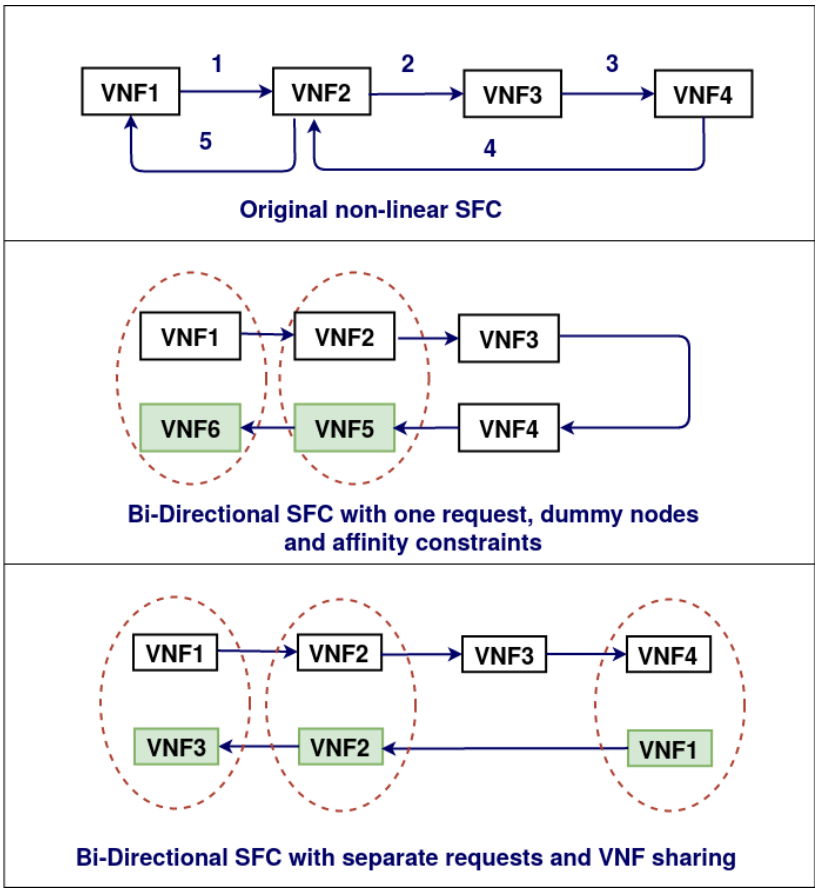


Figure 3.2: Request Adjustment for Non-Linear SFC Support

3.2.2 Non-Linear SFC Support

Furthermore, our model supports more complex non-linear SFCs where traffic flows through certain VNFs more than once, this scenario can be modeled in two ways as shown in Figure 3.2:

Separate SFC Request with VNF sharing

Each way of the SFC is modeled as a separate SFC request, while allowing the sharing of VNFs that are visited both ways, which can be required by specific applications (firewall allowing return traffic only for example). VNF sharing can also be generalized between different SFCs of the same slice, or different slices of the same tenant. However, if the SFC request of one way of the SFC is rejected, the second one should also be rejected.

One SFC Request with VNF duplication

If a VNF is visited more than once, it is duplicated by creating a *dummy* instance, and its link with the 'original' instance of the VNF is expressed using an attribute in the dummy node, pointing to the original VNF. Each VNF and its redundant *dummy* instances need to be mapped on the same location, which can be enforced using an affinity constraint. Furthermore, the amount of computing resources required by the redundant instances are null. Note that during the resource allocation phase, the required amount of resources for this type of VNFs needs to take into account the additional resource consumption that occurs when the flow is treated more than once by the same VNF.

3.2.3 Multi-Domain Placement and Request Partitioning

We provide the pseudo-code of the proposed Multi-Domain Placement procedure in Algorithm 1.

Initial Placement

When the multi-domain orchestrator receives a request req from a user to deploy an SFC, it calls the function **abstractPlacement** that performs an initial placement on the previously established abstract view of the topology $\mathcal{G}_{abs}$. This placement takes into account the QoS requirements defined by the SLA of the user, the placement constraints of each VNF such as affinity constraints, and the allowed locations. It should be noted that the optimization objectives and their levels of priority are also defined by the SLA.

Request Partitioning

Once the initial placement $abstractPl$ has been obtained, the SFC request is partitioned accordingly (See Figure 3.3), and dummy boundary nodes are added at the extremities of the sub-chains. These dummy nodes are required to be placed at the boundaries of the domains, to forward the traffic out of the domains and to the next sub-chain.

Algorithm 1: Multi-Domain Placement Algorithm

Input : Abstract Topology $\mathcal{G}_{abs}$, Request req
Set of Authorized Domains $\mathcal{M}$
Output: Vector containing the placement result

eval['Cost'] $\leftarrow$ 0; eval['Latency'] $\leftarrow$ 0 ;
absractPl $\leftarrow$ **abstractPlacement**(req,$\mathcal{G}_{abs}$,$\mathcal{M}$);
if *abstract placement failed* **then**
  return *False, null, null*
end
`// Perform request partitioning`
partReq $\leftarrow$ **requestPartitioning**(req,abstractPl);
for $p \leftarrow 0$ **to** |*partReqs*|*-1* **do**
  partialPl[p]$\leftarrow$ **localPlacement**(partReqs[p]);
  if *local placement fails* **then**
    `// Perform backtracking`
    if $\exists$*vnf* $\in \bigcup_0^p partReqs[n], |\mathcal{M}_{vnf}| > 1$ **then**
      find one vnf for which $|\mathcal{M}_{vnf}| > 1$;
      $\mathcal{M}_{vnf} \leftarrow \mathcal{M}_{vnf} - \{$abstractPl[vnf]$\}$;
      go to Step **1**;
    else
      return *False, null, null*
    end
  end
end
`// Compute E2E cost & latency`
for *i* $\in$ *['Cost','Latency']* **do**
  $eval[i] \leftarrow \sum_{p \leftarrow 0}^{|partReqs|-1}$ partPl[p][i] $+ \sum_{n \leftarrow 1}^{|\mathcal{M}|-1} \mathcal{G}_{abs}$[absPl[n-1]][absPl[n]]$[i]$;
end
send placement confirmation to local domains;
update $\mathcal{G}_{abs}$;
return *[True,eval]*

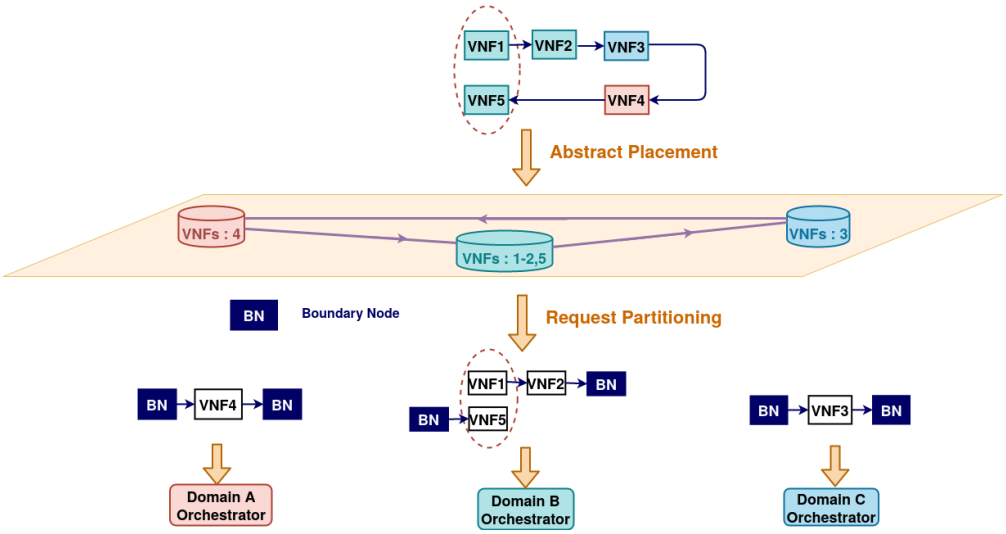


Figure 3.3: SFC Request Partitioning

Local Placement

Next, the sub-requests $partReqs$ are sent to the chosen domains in order to perform a local placement **localPlacement** as shown in Figure 3.4, while having a full view of their respective local topologies. Note that each local orchestrator can implement their own placement algorithm. If the placement of a sub-chain fails, a backtracking mechanism is prompted in order to perform the initial placement once again, but while ruling out the previous placement solution. Once the multi-domain orchestrator receives the placement results of all of the domains, it computes the end-to-end cost and latency of the SFC, then sends a confirmation to the local domain orchestrators in order to start the sub-SFCs deployment.

3.3 Problem Formulation

We present in this section our ILP formulation of the placement problem. Table 3.1 summarizes the notations that have been used i the formulation.

Notation	Description
Sets	
$\mathcal{S}$	Set of SFCs
$\mathcal{V}_i$	Set of VNFs in SFC i
$\mathcal{A}_i/\bar{\mathcal{A}}_i$	Set of affinity/anti-affinity constraints for SFC i
$\mathcal{M}_{i,j}$	Set of allowed placement nodes for VNF j in SFC i
$\Re$	Set of computing resources (CPU, RAM, disk)
$\mathcal{N}_d$	Set of nodes in domain d
$\mathcal{L}$	Set of links
Decision Variables	
$\mathcal{X}_{i,j}^{n}$	Placement of VNF j of SFC i on node n (Boolean)
$\mathcal{X}_{i,j,j+1}^{n,m}$	Placement of VNF i of SFC i on node n and its successor on node m (Boolean)
$\mathsf{Z}_{i,j,j+1}^{n,m,q}$	Placement of link between VNFs j and $j+1$ of SFC i on qth physical link between nodes n and m (Boolean)
Request	
$\nabla_{r,i,j}$	Required amount of computing resource r for VNF j of SFC i
$\mathcal{W}_{i,j,j+1}$	Required bandwidth for the virtual link between VNFs j and $j+1$ of SFC i
Network Infrastructure	
$\rho_{i,n,m}$	Number of physical paths between the nodes n and m that are allowed for SFC i
$\mathcal{R}_{r,n}$	Amount of computing resource r available on node n
$\zeta_{r,n}$	Cost per unit of resource r on node n
$\tau_l^{n,m,q}$	Link l is part of the qth path between nodes n and m (Boolean)
$\mathcal{R}_{\omega,l}$	Capacity of link l
ϕ_l	Latency of link l
ζ_l	Cost of using the link l per bandwidth unit

Table 3.1: Notations used

3.3.1 Network Architecture

The network infrastructure is modeled as a weighted undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \mathcal{N}_d \cup \mathcal{N}_s$ is the set of vertices representing the forwarding devices (switches and routers) $\mathcal{N}_s$, and the Data Center nodes $\mathcal{N}_d$ with associated computation, memory and storage capacities; the vertices are connected by edges representing physical links from the set $\mathcal{L}$, that are characterized by bandwidth capacity, as well as the induced latency. We will also denote by $\mathcal{P}$ the set of pre-determined physical paths between the physical nodes. We denote by $\mathcal{S}$ the set of SFCs, and by $\mathcal{V}_i$ the set of VNFs in the SFC i.

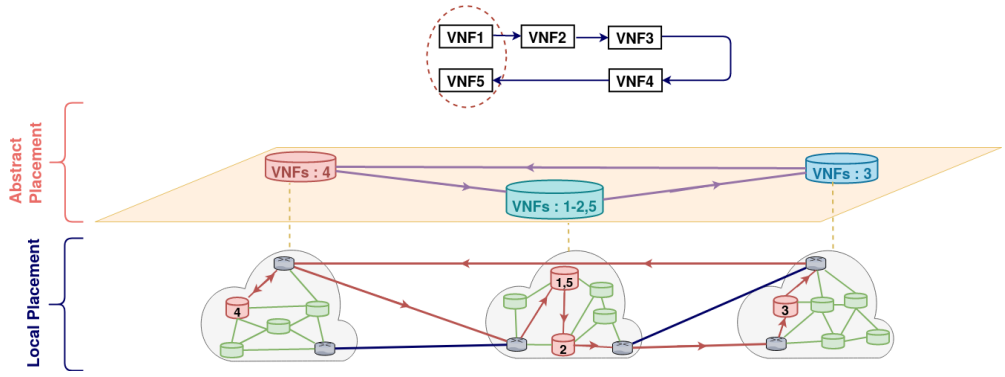


Figure 3.4: End-to-End SFC Placement

3.3.2 SFC Request Formulation

Each SFC request is modeled as a directed weighted graph. VNFs are represented by the vertices with associated resource requirements and a set of authorized nodes $\mathcal{M}_{i,j}$ on which the VNFs can be placed. They are connected by the graph's edges, with associated link requirements. We specify bandwidth requirements for each individual link, as some VNFs may change SFC's traffic volumes. The requests also specify the user's preferences regarding the optimization objectives (i.e. cost and latency) according to their SLA. In our model, we enable non-linear SFC placement using *dummy* nodes that are bound to the original VNFs using affinity constraints in order to route the traffic back to these particular VNFs; we assume that the amount of computing resources allocated to each VNF has been adjusted to the total volume of processed traffic during the resource allocation phase, which is out of the scope of this work.

3.3.3 Constraints

Placement Constraints

Node Mapping The boolean variable $\mathcal{X}_{i,j}^{n}$ expresses whether the j^{th} VNF of the SFC i has been mapped to the physical node n.

$$\mathcal{X}_{i,j}^{n} = \begin{cases} 1 & \text{If VNF } j \text{ of SFC } i \text{ is placed on node } n \\ 0 & \text{Otherwise} \end{cases}$$

$$\sum_{n \in \mathcal{M}_{i,j}} \mathcal{X}_{i,j}^{\text{n}} = 1, \qquad i \in \mathcal{S}, \forall j \in \mathcal{V}_i \tag{3.1}$$

$$\mathcal{X}^{\text{n}}_{i,k_1} - \mathcal{X}^{\text{n}}_{i,k_2} = 0, \qquad i \in \mathcal{S}, \mathcal{A}_{i,j} \subset \mathcal{A}_i, k_1, k_2 \in \mathcal{A}_{i,j}, n \in \mathcal{N}_d \tag{3.2}$$

$$\mathcal{X}^{\text{n}}_{i,k_1} + \mathcal{X}^{\text{n}}_{i,k_2} \leq 1, \qquad i \in \mathcal{S}, \bar{\mathcal{A}}_{i,j} \subset \bar{\mathcal{A}}_i, k_1, k_2 \in \mathcal{A}_{i,j}, n \in \mathcal{N}_d \tag{3.3}$$

$i \in \mathcal{S}, \forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_j, \forall m \in \mathcal{M}_{j+1}$:

$$\mathcal{X}^{\text{n}}_{i,j} \cdot \mathcal{X}^{\text{m}}_{i,j+1} \leq \rho_{i,n,m}, \tag{3.4}$$

Constraint 3.1 ensures that each VNF is mapped to only one of its authorized physical nodes. Constraints 3.2 and 3.3 express affinity/co-location (i.e. two VNFs must be placed on the same node) and anti-affinity/anti-location requirements (i.e. two VNFs must not be placed on the same node), where $\mathcal{A}_i$ and $\bar{\mathcal{A}}_i$ represent the set of affinity and anti-affinity constraints for SFC i respectively. Each set comprising tuples of VNFs k_1 and k_2 for which the constraints apply. We denote by $\rho_{i,n,m}$ the number of physical paths between the nodes n and m that are allowed for SFC i, constraint 3.4 ensures that two consecutive VNFs are mapped to two nodes if and only if there is at least one allowed physical path between these nodes. Constraint 3.4 is quadratic, it can be linearized by introducing a new boolean variable $\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1}$ which is the product of $\mathcal{X}^{\text{n}}_{i,j}$ and $\mathcal{X}^{\text{m}}_{i,j+1}$:

$$\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1} = \mathcal{X}^{\text{n}}_{i,j} \cdot \mathcal{X}^{\text{m}}_{i,j+1} \tag{3.5}$$

Constraint 3.4 is therefore translated to the following:

$$\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1} \leq \mathcal{X}^{\text{n}}_{i,j} \tag{3.6}$$

$$\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1} \leq \mathcal{X}^{\text{m}}_{i,j+1} \tag{3.7}$$

$$\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1} \geq \mathcal{X}^{\text{n}}_{i,j} + \mathcal{X}^{\text{m}}_{i,j+1} - 1 \tag{3.8}$$

$$\dot{\mathcal{X}}^{\text{n,m}}_{i,j,j+1} \leq \rho_{i,n,m} \tag{3.9}$$

Link Mapping We will denote by $\mathcal{P}^{n,m,q}$ the q^{th} physical path between nodes n and m. The boolean variable $\mathcal{Z}^{n,m,q}_{i,j,j+1}$ determines whether the logical link between the j^{th} VNF and its successor in the SFC i has been mapped to the q^{th} physical path between nodes n and m.

$$\mathcal{Z}^{n,m,q}_{i,j,j+1} = \begin{cases} 1 \text{ If logical link between VNFs } j \text{ and } j+1 \text{ of SFC } i \\ \quad \text{is mapped to the } q^{\text{th}} \text{ physical path between nodes } n \text{ and } m \\ 0 \text{ Otherwise} \end{cases}$$

$\forall i \in \mathcal{S}, \forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_j, \forall m \in \mathcal{M}_{j+1}$:

$$\sum_{q=1}^{\rho_{n,m}} \mathcal{Z}^{n,m,q}_{i,j,j+1} = \mathcal{X}^{\text{n}}_{i,j} \cdot \mathcal{X}^{\text{m}}_{i,j+1} \tag{3.10}$$

Constraint 3.10 ensures that a logical link between two VNFs is mapped to a physical path if and only if the corresponding VNFs are mapped to the nodes that the path interconnects, and vice versa; it also ensures that not more than one physical path is allocated to a logical link. Similarly to 3.4, the constraint can be linearized as follows:

$$\sum_{q=1}^{\rho_{n,m}} \mathcal{Z}_{i,j,j+1}^{n,m,q} = \dot{\mathcal{X}}_{i,j,j+1}^{\mathrm{n,m}} \tag{3.11}$$

Capacity Constraints

Computing resources We will denote by $\Re$ the set of computing resource types of physical nodes (CPU, RAM, disk space...), by $\nabla_{r,i,j}$ the required amount of the resource r for the jth VNF of the ith SFC, and by $\mathcal{R}_{r,n}$ each node n's remaining capacity for the resource type r. Constraint 3.12 ensures that the total amount of allocated resources on each node for each resource type does not exceed the amount of available resources remaining on the node.

$$\sum_{j \in \mathcal{V}_i} \nabla_{r,i,j} \cdot \mathcal{X}_{i,j}^{\mathrm{n}} \leq \mathcal{R}_{r,n}, \qquad \forall n \in \mathcal{N}_d, \forall r \in \Re \tag{3.12}$$

Link resources We will denote by $\mathcal{W}_{i,j,j+1}$ the required bandwidth for the virtual link between the jth VNF and its successor, and by $\mathcal{R}_{\omega,l}$ each link l's capacity; we will also use the boolean $\tau_l^{n,m,q}$ to express whether the link l is part of the qth path between the nodes n and m. Constraint 3.13 ensures that the total allocated bandwidth on each physical link of an end-to-end physical path does not exceed its remaining capacity.
$\forall i \in \mathcal{S}, \forall l \in \mathcal{L}$:

$$\sum_{j \in \mathcal{V}_i} \sum_{n \in \mathcal{M}_j} \sum_{m \in \mathcal{M}_{j+1}} \sum_{q=1}^{\rho_{n,m}} \mathcal{W}_{i,j,j+1} \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \leq \mathcal{R}_{\omega,l} \tag{3.13}$$

3.3.4 Objective Function

The objective of our model is to minimize the overall cost, as well as the end-to-end latency. However, these objectives are contradictory: the cost of the physical links is inversely proportional to their latency (i.e. lower latency links are more expensive). A trade-off is then obtained by setting weights to both objectives in order to set optimization priorities according to the user's preferences as specified in the SLAs (uRLLC, best effort, etc.). Therefore, our optimization objective function is a weighted sum of normalized values of the overall cost and the end-to-end latency for each SFC. Let us define the following parts of the objective function:

$$\mathcal{C}_{computing} = \sum_{n \in \mathcal{N}_d} \sum_{r \in \Re} \sum_{j \in \mathcal{V}_i} \zeta_{r,n} \cdot \nabla_{r,i,j} \cdot \mathcal{X}^{\mathrm{n}}_{i,j} \tag{3.14}$$

$$\mathcal{C}_{link} = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_j} \sum_{m \in \mathcal{M}_{j+1}} \sum_{q=1}^{\rho_{n,m}} \zeta_l \cdot \mathcal{W}_{i,j,j+1} \cdot \mathcal{Z}^{n,m,q}_{i,j,j+1} \cdot \tau^{n,m,q}_l \tag{3.15}$$

$$\phi = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_j} \sum_{m \in \mathcal{M}_{j+1}} \sum_{q=1}^{\rho_{n,m}} \phi_l \cdot \mathcal{Z}^{n,m,q}_{i,j,j+1} \cdot \tau^{n,m,q}_l \tag{3.16}$$

Equations 3.14 and 3.15 express computational and link costs for the SFC i respectively, with $\zeta_{r,n}$ being the cost of using the resource r on node n per unit, and ζ_l the cost of using the link l per bandwidth unit. Equation 3.16 expresses the end-to-end latency for the SFC i, with ϕ_l being the link l's latency. The end-to-end latency is computed as the sum of the latencies of the used links, while omitting the processing time of the VNFs. Indeed, their values mainly depend on the computations that are performed by each VNF, as well as the amount of resources that are assigned to the VNF. Therefore, we suppose that these times aren't impacted by the SFC placement, and can be discarded in this model. The optimization objective is to minimize the following objective function while satisfying the aforementioned constraints.

$$\textbf{minimize} \qquad \alpha_{\mathcal{C}} \cdot \frac{\mathcal{C}_{computing} + \mathcal{C}_{link}}{\lambda_{\mathcal{C}}} + \alpha_{\phi} \cdot \frac{\phi}{\lambda_{\phi}}$$

subject to

$$\begin{cases} \text{Constraints 1 - 3} \\ \text{Constraints 6 - 9} \\ \text{Constraints 11 - 13} \end{cases}$$

With $\alpha_{\mathcal{C}}$ and α_{ϕ} being the weights associated to cost and latency respectively, which are determined from the users' SLAs; and $\lambda_{\mathcal{C}}$ and λ_{ϕ} the normalization coefficients for cost and latency, which are determined by computing the mean cost and latency for the topology.

3.4 Heuristic

Although an exact solution for the previously detailed ILP model ensures an optimal placement of SFC requests considering the provided information and constraints, it lacks scalability. Indeed, optimal SFC placement has been proven to be an NP-Hard problem [78], which means that computation time increases exponentially on large problem instances as illustrated in our evaluation results in Section 4.6, thus making this solution impractical for operational use. However the ILP

model's optimal solution can be used as a reference in order to evaluate the efficiency of alternative solutions.

As an alternative, and in order to support bigger instances of the problem, we propose a memetic heuristic based on a genetic algorithm, combined to a local search method for solution improvement as shown in Algorithm 2. At first, a set of initial solutions is generated, evaluated, and classified according to the results of the objective function; the set is then updated at each generation by introducing new individuals obtained through solution generation, mutation, crossover, or local search improvement. Afterwards, the set of solutions is evaluated and classified using the objective function previously expressed in the ILP, and the best individuals are selected for the next generation. The process is repeated until a fixed number of generations is reached. The best solution is then returned. Note that all of the operators use intervals in order to reduce request fragmentation. The main steps of our heuristic are described in the following, and illustrated in Figure 3.5.

Solution Generation New individuals are generated as follows: the placement of the first VNF is randomly determined from the list of allowed locations, then each subsequent VNF is also placed on the same location as long as the VNFs' affinity, placement constraints, and node's capacity allow it, thus reducing link-associated cost and latency. Route mapping is performed by choosing the best available path according to the request's optimization objectives' weights.

Mutation During the mutation phase, a solution is randomly selected, then the placement and chaining of a random interval of successive VNFs is changed while respecting the aforementioned constraints.

Crossover Two solutions are selected at random, and are crossed at a random interval (the placement of the VNFs in this inteval are swapped between the two solutions) in order to generate a new solution, and VNF chaining is also updated.

Local Search Improvement In this phase, a random solution is selected, and its neighbor solutions are explored in order to operate local improvements (e.g. move one or more VNFs to a nearby node).

Algorithm 2: Proposed Placement Heuristic

Input : Local Topology $\mathcal{G}$, Request req
Set of Authorized Nodes $\mathcal{M}$
Output: Placement pl, cost, latency

```
Generate initial individuals ;
repeat
    failed ← 0 ;
    repeat
        operation=random(0,3) ;
        switch the value of operation do
            case 0 do Select 2 random individuals;
            sol ← Crossover at a random interval ;
            case 1 do Select random individual ;
            sol ← Mutation for a random interval;
            case 2 do sol ← Individual generation;
            case 3 do Select random individual ;
            sol ← Local search improvement;
        end
        failed ← failed + 1 ;
        if failed = Threshold then
            return False, null, null
        end
    until sol[0] = True;
    Compute solution fitness according to SLA;
    Select the best individuals for next generation;
until Number of generations is reached;
placement ← Get the best individual from set;
cost , latency ← Get end to end cost and latency
                 of selected placement;
return True, cost, latency
```

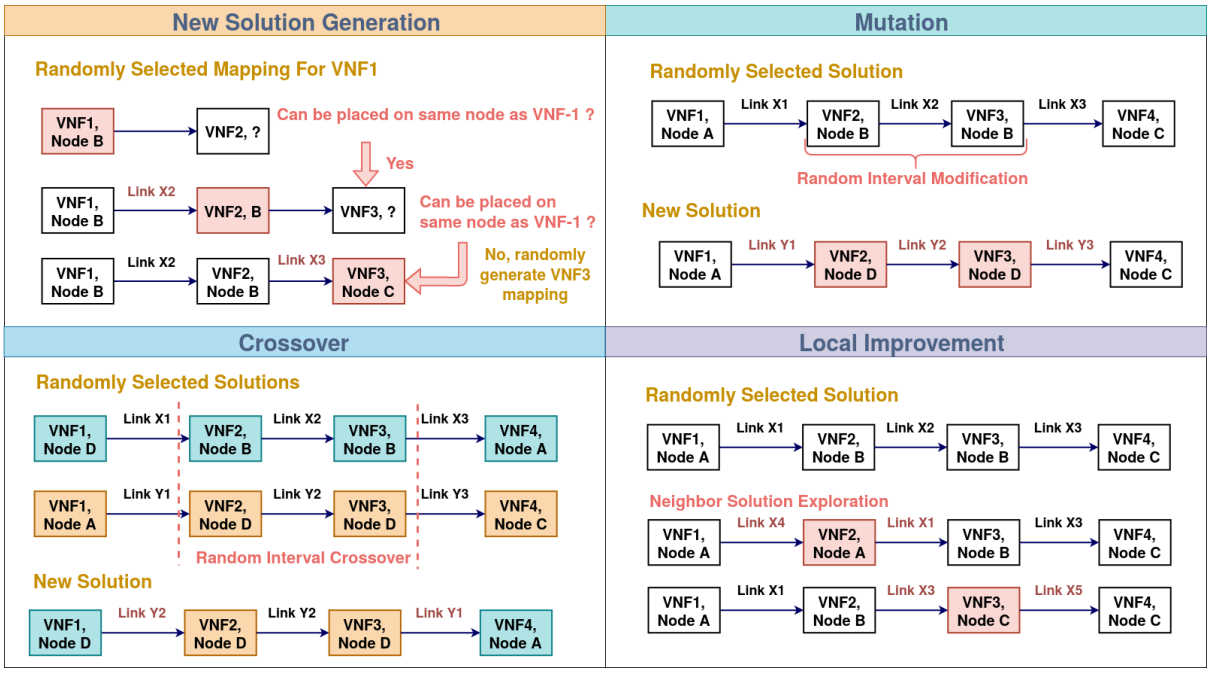


Figure 3.5: Solution Generation Steps of the Proposed Heuristic

3.5 Evaluation

3.5.1 Simulation Environment

We evaluated our solution by conducting simulations on the hardware and software configuration outlined in Table 3.2.

Component	Configuration
CPU	Intel Xeon E5-2640 v3 (32 Cores), $2.60GHz$
RAM	$128GB$
OS	Ubuntu 16.04, 64 bits

Table 3.2: Testbed Hardware and Software Configuration

The simulation program and heuristic were developed using the Python programming language; we also used the Gurobi [147] optimizer in order to find the optimal solution for our ILP model.

For the test environment, we generated multi-domain topologies using the *networkx* [148] library. Each domain comprises massively connected nodes via intra-domain links where each domain comprising n nodes, has $n * (n-2)/2$ edges; and is able to communicate with the other domains via edge routers connected through WAN links. The type of nodes is randomly selected (Small, Medium, Large), where each type is equivalent to a certain amount of CPU, RAM, and storage resource capacity, with different costs per resource unit, similarly, the inter-domain links can be Low, Medium, or High latency, with different costs per bandwidth unit. We also pre-computed the set of paths between nodes for each domain, with a path length limited to 3 hops. After each placement iteration, the graph's resource capacity values are updated, by deducing the amount of consumed link and computing resources from the available capacity of each link and node that have been used. We evaluate our solution using 3 network topologies of different sizes, as detailed in Table 3.3.

Features \ Topology	Small	Medium	Large
Number of Domains	4	6	8
Number of Nodes per Domain	10	15	30
Number of Links	166	603	3388

Table 3.3: Topology Information

Requests are randomly generated and placed sequentially. Each request specifies the amount of required resources and a set of allowed placement locations for each VNF and link, as well as the dependencies between VNFs, and optimization objective's associated weights (cost and latency) obtained from the SLA class of the client. For our simulations, we construct the SFC requests using 5 VNF types (small, medium, large, dummy, boundary) with corresponding resource and placement requirements, and 5 SLA classes (ultra low latency, low latency, a fair trade-off between cost and latency, low cost, best effort), with different levels of priority for our optimization objectives, as shown in Table 3.4. The solution is evaluated for SFC lengths ranging from 4 to 10 in the small and medium topologies, and from 4 to 8 in the large topology.

SLA	Cost weight	Latency Weight
Ultra Low Latency	0.1	0.9
Low Latency	0.2	0.8
Equilibrium	0.5	0.5
Low Cost	0.8	0.2
Best Effort	0.9	0.1

Table 3.4: Weights assigned to each objective depending on SLA

3.5.2 Methodology

The simulations for each configuration are run 20 times. At each iteration, the placement algorithms are run for 150 successive requests for the small and medium topologies, and 50 requests for the large one. In this evaluation, SFC placement is performed using the ILP with full knowledge of the network topology, as well as the proposed heuristic combined to the multi-domain hierarchical scheme with a limited view on the infrastructure. We vary the number of generations (iterations) of our heuristic in order to evaluate its effect on efficiency and computational time, the number of generations depends on each configuration's parameters, namely the topology size and SFC length: $N_1 = |req| * |\mathcal{G}|$, $N_2 = \frac{N_1}{4}$, $N_3 = \frac{N_1}{16}$. The placement results of the different solutions are evaluated using two key metrics:

- Placement efficiency, which can be obtained using the aforementioned objective function, by computing the ratio $\frac{f(sol_{ILP})}{f(sol_h)}$ between the results of the evaluation function for the heuristic solution $f(sol_h)$, and the ILP solution $f(sol_{ILP})$. This ratio then expresses how close the heuristic solution is to the ILP (optimal) one, if the heuristic solution provides an optimal solution, the ratio would have a value of 1, and this value decreases when the heuristic solution provides sub-optimal solutions.
- Scalability, which can be quantified by measuring the algorithm's computational time.

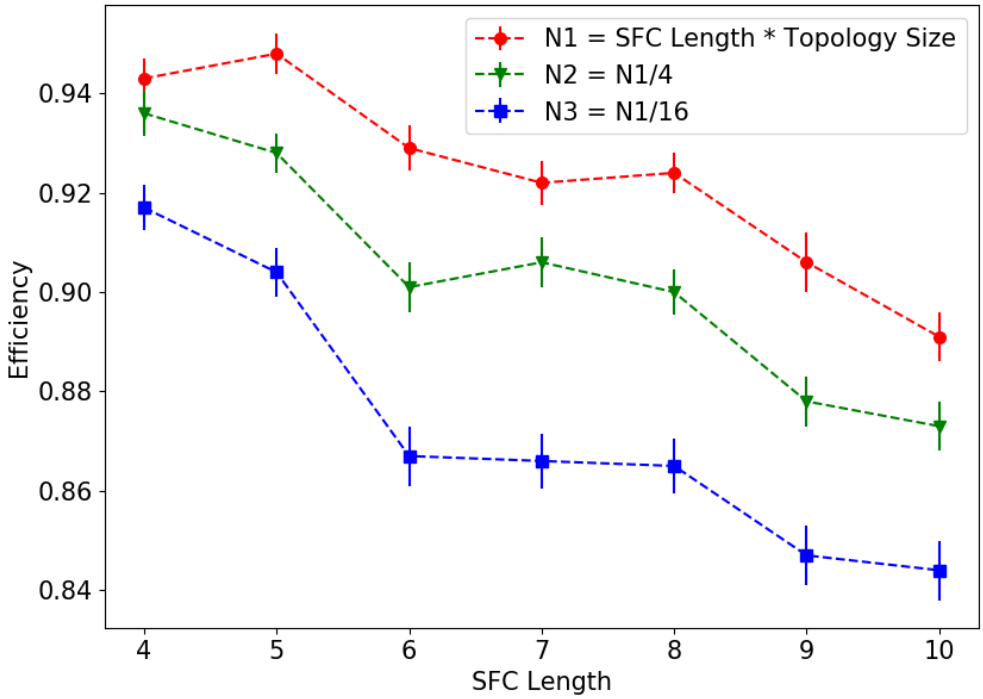


Figure 3.6: Relative efficiency of the heuristic solution for the Small Topology depending on the number of generations N

3.5.3 Results

Efficiency

Figures 3.6, 3.7, and 3.8 illustrate the mean values and 95% Confidence Interval (CI) of the efficiency of our heuristic compared to the exact solution using different topologies and SFC sizes. Efficiency is measured by computing the average ratio between the results of the evaluation of the ILP placement and those obtained from the heuristic placement for each request.

We can observe that although the heuristic does not dispose of the full view of the network, nor does it explore the full set of solutions, it displays an accuracy of $94.3 - 89.1\%$ with a 95% CI of $0.8 - 1\%$ in the small network as illustrated in Figure 3.6, and $96.5 - 92.2\%$ with a CI of $0.6 - 0.9\%$ in the medium network as shown in Figure 3.7 when the number of generations is fixed to N_1. Whereas in the large topology (Figure 3.8), the heuristic's efficiency decreases to $96.9 - 89.8\%$ with a CI of $1 - 3\%$ for the number of generations N1.

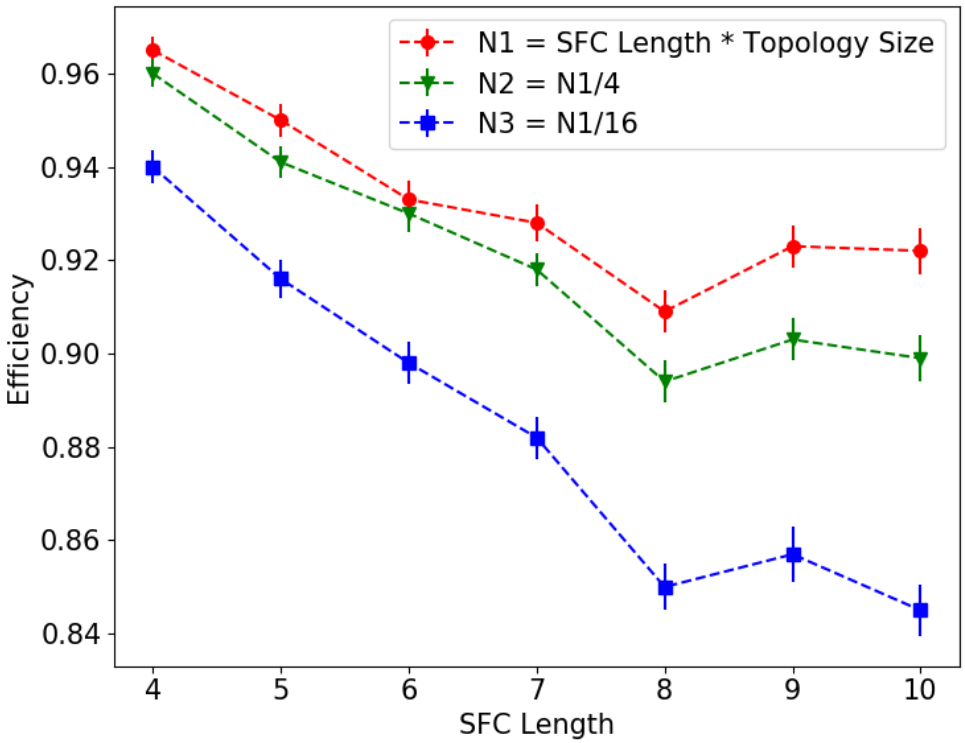


Figure 3.7: Relative efficiency of the heuristic solution for the Medium Topology depending on the number of generations N

Increasing the SFC length also causes a decrease in efficiency. We can notice a reduction of $4-6\%$ when increasing the SFC length for numbers of generations N_1 and N_2 regardless of the topology size. Indeed, efficiency merely decreases from 94.3% and 93.6% to 89.1% and 87.3% respectively in the small topology, from 96.5% and 96% to 92.2% and 89.9% respectively in the medium topology, and from 96.9% and 96.6% to 90.1% and 87.4% respectively in the large topology. When setting the number of generations to N_3, we notice that increasing SFC length has a more significant impact on efficiency as it is reduced by $7-11\%$, reaching a ratio of 84.4% with a CI of 1.2% in the small network, 84.5% with a CI of 1.1% in the medium topology, and 81.7% with a CI of 3.2% in the large one. In contrast, for small SFC lengths, decreasing the number of generations does not have a significant effect on efficiency as it remains above 90% regardless of the topology.

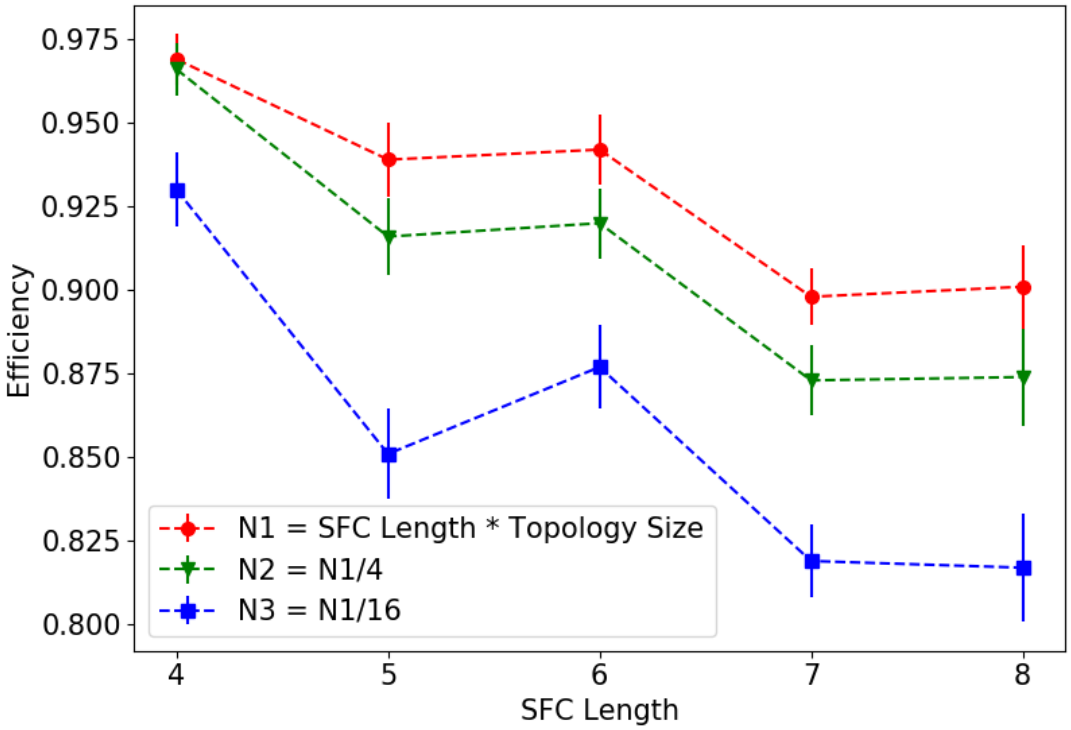


Figure 3.8: Relative efficiency of the heuristic solution for the Large Topology depending on the number of generations N

Scalability

Table 3.5 features the mean computation times for the ILP and the heuristic solutions when varying the topology size and the SFC length, as well as the number of generations for the heuristic. The exact solution demonstrates its lack of scalability as it takes up to many hours in order to produce a solution in the large network case as well as the medium-sized network when the SFC length is increased, which makes it impractical for operational deployment. In contrast, the heuristic proves its scalability as it provides solutions in realistic times regardless of the topology and SFC size: $81ms$ and up to a second in the small topology, $422ms$ to less than 4 seconds in the medium network, and $2.94s$ to $65s$ in the large network when the number of generations is fixed to N_2, which is an improvement of up to 600 times compared to the ILP. When setting the number of generations to N_1, execution time increases to a range between $241ms$ and $3.46s$ in the small topology, between $1.32s$ and $13s$ in the medium topology, and between $10s$ and $222s$ in the large one. Com-

putational time can be further reduced by setting the number of generations of the heuristic to N_3. In this case, it ranges between $37ms$ and $429ms$ in the small topology, $166ms$ and $1440ms$ in the medium topology, and between $994ms$ and $19s$ in the large topology depending on SFC length.

		SFC Length						
Topology	Method	4	5	6	7	8	9	10
Small	ILP	1.48	2.14	3.09	5.38	7.68	11.59	17.05
	N1	0.241	0.350	0.631	1.03	1.25	2.22	3.46
	N2	0.081	0.113	0.201	0.322	0.391	0.678	1.06
	N3	0.037	0.052	0.088	0.138	0.167	0.278	0.429
Medium	ILP	16.29	27.74	103	224	376	1016	2623
	N1	1.32	1.93	3.16	4.43	7.72	10.02	13.94
	N2	0.422	0.588	0.938	1.30	2.21	2.88	3.98
	N3	0.166	0.233	0.361	0.491	0.819	1.06	1.44
Large	ILP	1381	2234	3468	6434	7364	-	-
	N1	10.25	27.81	69.31	123	147	154	222
	N2	2.94	7.92	19.47	33.55	40.57	42.13	65.47
	N3	0.994	2.72	6.51	11.71	12.63	13.12	19.76

Table 3.5: Comparison of computation times (in seconds)

3.5.4 Results Discussion

In light of the obtained results, three main observations can be made:

- The heuristic associated with the multi-domain SFC placement and partitioning framework, with limited visibility, provides close to optimal results with $96.5 - 89.1\%$ of efficiency compared to the formulated ILP with full visibility on the network infrastructure. Furthermore, these results are obtained in significantly lower computational times, with a reduction of up to 400 times.

- The heuristic's efficiency is mainly affected by the length of the SFC, as well as the number of generations/iterations. Indeed, the increase of the SFC length, as well as the decrease in the number of generations cause a degradation in the heuristic's efficiency. However, topology size doesn't seem to have a negative impact. In fact, the heuristic's efficiency increases with topology size, which could be explained by the fact that a larger topology

provides a larger choice of solutions, thus making it easier to find one or more optimal solutions.

- The computational times for the heuristic are affected by SFC length, topology size, as well as the number of generations. Indeed, the increase of any of these parameter leads to an increased number of iterations for the heuristic and therefore a longer execution time, as the first two parameters also affect the number of generations of the heuristic. However, while the ILP's computational times increase exponentially, the heuristic's remain in the range of seconds for the biggest problem instances.

We can therefore conclude that the heuristic is sufficiently efficient and scalable for use in different configurations. Furthermore, the number of generations for which the algorithm is run can be tuned in order to find a satisfactory trade-off between computational cost and closeness to the optimal placement solutions.

3.6 Conclusion

In this chapter, we proposed a multi-objective solution for joint node and link mapping of multi-domain SFCs using an ILP model, as well as a memetic algorithm. The results of our heuristic with limited visibility on the network are close to the optimal one by $96.5 - 89.1\%$ with a computational time improvement of up to 400 times. However, the weighted sum approach doesn't accurately reflect the preferences of the user, and some objectives might be optimized at the expense of the others. Furthermore, an iterative process is required in order to set the weights associated with each objective.

In the next chapter, we develop a solution that allows a finer grained specification of user preferences using Physical Programming.

Chapter 4

SFC Placement - Physical Programming

4.1 Introduction

In the context of 5G and its use cases, the SFC placement process requires the optimization of different QoS metrics such as bandwidth, where a minimum value needs to be guaranteed, and latency, that needs to be kept at a minimum. Furthermore, the deployment cost needs to be minimized in order to maximize profit. Therefore, the SFC placement process is a Multi-Objective Optimization problem. Multi Objective Optimization (MOO) is a process where more than one optimization objectives are considered simultaneously. This process requires the articulation of the Decision Maker's (DM) preferences regarding the objectives, which will influence the priority that each objective would have over the others in the optimization process. Many methods have been proposed in order to express the preferences of the DM before or after the optimization. Most of these methods rely on the assignment of *weights* to each objective, and the optimization of the weighted sum of these objectives [149]. However, these weights are generally determined through trial and error, and do not reflect the specific preferences of the DM as they are not physically meaningful. Furthermore, as each objective has a different scale, a normalization process is required.

In this chapter, we dive deeper in the multi-objective optimization aspect by proposing a more detailed formulation that enables the expression of the user's preferences in terms of meaningful physical values, using Physical Programming.

4.2 Physical Programming

The Physical Programming Optimization approach was proposed by Messac *et al.* in its Non-Linear [150] and Linear [151] forms. It allows the use of physically meaningful parameters in order to express preferences for each objective using preference ranges, and different classes that can be hard (H) or soft (S), as shown in Figure 4.1:

- **Hard Classes:** These classes are constraints by definition, because solutions are rejected if they are not in the acceptable range.
- **Soft Classes:** As illustrated in Figure 4.2, these soft classes objective functions include 6 preference ranges: *Highly desirable*, where improvement past the ideal value is of minimal additional value, *Desirable*, *Tolerable*, *Undesirable*, *Highly Undesirable*, and *Unacceptable*, which is expressed using a constraint.

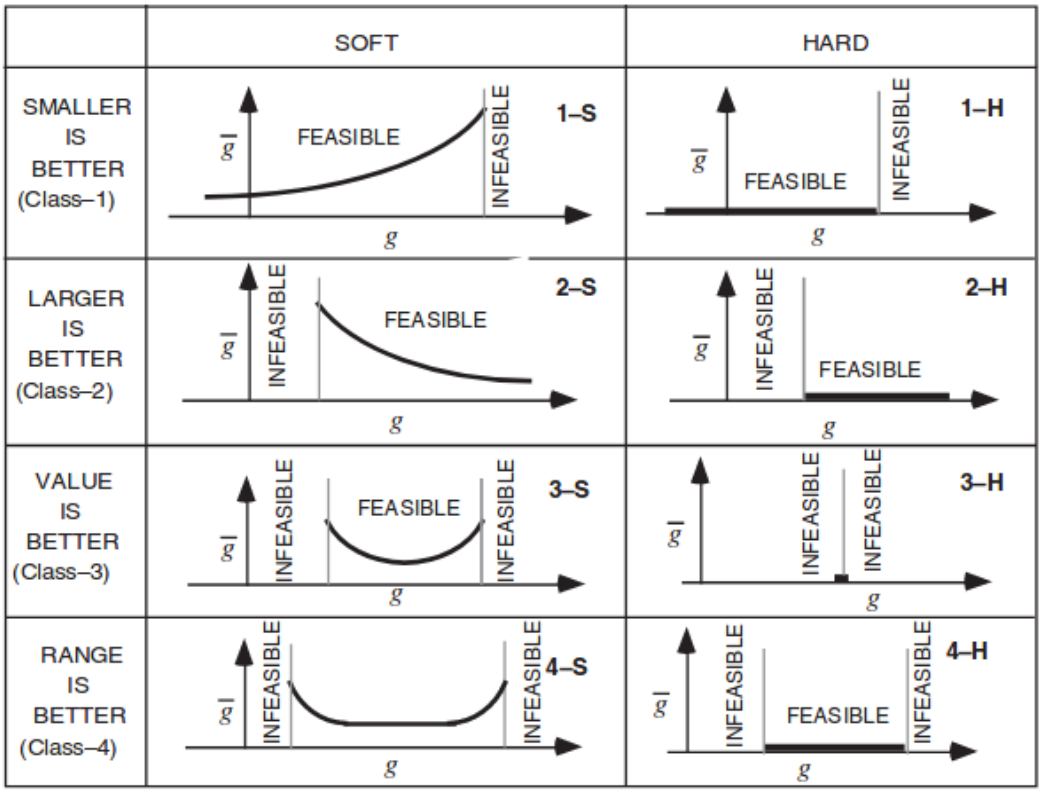


Figure 4.1: Physical Programming Preference Classes

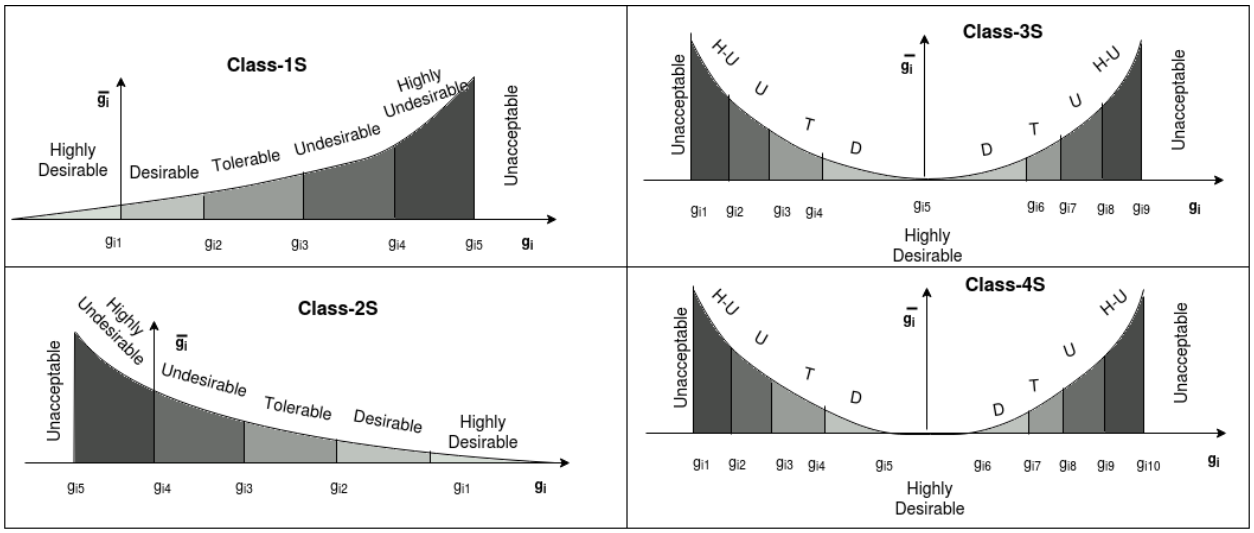


Figure 4.2: Region Boundaries For Preference Classes

Multiple methods have been proposed for Physical Programming: Linear Physical Programming [151] (LPP), Non-Linear Physical Programming (NLPP) [150], Global Physical Programming (GPP) [152], and Fuzzy Physical Programming (FPP) [153]. Each method expresses the objective function for each criterion as a combination of piece-wise convex functions (one function for each range) using the values of the range boundaries, and the global objective function is defined as the sum of these functions. The approaches for obtaining the objective functions for each Physical Programming method are explained in more details in Section 4.3.2.

Physical Programming optimization enforces the One vs Others Criteria (OVO) Rule, which states that a full reduction for one criterion across a given range is better than a full reduction for all of the other criteria across the next better range. In practice, it means that between the options of improving on criterion from the Tolerable range to the Desirable range, and improving all the other criteria from the Desirable to the Highly Desirable range, the first option would be preferred. This ensures a fair trade-off between objectives, and avoids improving certain objectives at the expense of others, as could be the case with the weighted method. This rule also ensures that the value of the objective function for range boundaries is the same across criteria (and therefore that each region's objective function's amplitude is also the same across criteria), which has a normalizing effect. Furthermore, once a certain objective has reached the ideal value, instead of further improving it past the ideal value, the focus would be on the remaining objectives.

A few works have leveraged on the Physical Programming method for Multi-Objective Optimization in different fields [154]: aircraft parameter design [155] [156], trajectory planning [157], mechanical engineering [158], as well as electromagnetic and transmission related problems [159] [160]. The results obtained with Physical Programming in each of the aforementioned works are closer to the user's preferences than the results obtained with the usual MOO methods.

4.3 Problem Formulation

We formulate in the following the model of the multi-domain SFC placement problem using Physical Programming, on both levels: the initial placement on the abstracted view, and the local placement performed by the domains. We first define the different placement, resource, latency, and cost constraints. Then, we express the objective function using Linear, Non-Linear, and Global Physical Programming. Table 4.1 features the notations that are used in our model.

4.3.1 Constraints

In this formulation, we use the same placement constraints, and capacity constraints for the computing resources as in the ILP formulation in the previous chapter in Section 3.3. However, the constraints for link resources are updated in order to support variable bandwidth values, and constraints related to bandwidth, relative cost, as well as the end-to-end latency are added.

Link resources We designate by $\mathcal{U}_i$ the number of users that the SFC i should be able to serve, and by $\mathcal{R}_{\omega,l}$ each link l's capacity; we will also use the boolean $\tau_l^{n,m,q}$ to express whether the link l is part of the qth path between the nodes n and m or not. We will denote by $\mathcal{W}_i$ the allocated bandwidth per user for the SFC i, depending on the SLA of the request, $\mathcal{W}_i$ could be a fixed value for the strict SLAs, or variable in cases of SLAs with more relaxed constraints, with a minimal value for the bandwidth. We can therefore discern two different cases:

- **The bandwidth value is fixed:** $\mathcal{W}_i$ is a constant. Constraint 4.1 ensures that the total allocated bandwidth on each physical link of an end-to-end physical path does not exceed its remaining capacity.
  $\forall i \in \mathcal{S}, \forall l \in \mathcal{L}$:

$$\sum_{j \in \mathcal{V}_i} \sum_{n \in \mathcal{M}_{i,j}} \sum_{m \in \mathcal{M}_{i,j+1}} \sum_{q=1}^{\rho_{n,m}} \mathcal{W}_i \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \cdot \mathcal{U}_i \leq \mathcal{R}_{\omega,l} \tag{4.1}$$

Notation	Description
Sets	
$\mathcal{S}$	Set of SFCs
$\mathcal{V}_i$	Set of VNFs in SFC i
$\mathcal{A}_i$/$\bar{\mathcal{A}}_i$	Set of affinity/anti-affinity constraints for SFC i
$\mathcal{M}_{i,j}$	Set of allowed placement nodes for VNF j in SFC i
$\Re$	Set of computing resources (CPU, RAM, disk)
$\mathcal{N}_d$	Set of nodes in domain d
$\mathcal{L}$	Set of links
Decision Variables	
$\mathcal{X}_{i,j}^{n}$	Placement of VNF j of SFC i on node n (Boolean)
$\dot{\mathcal{X}}_{i,j,j+1}^{\text{n,m}}$	Placement of VNF i of SFC i on node n and its successor on node m (Boolean)
$\mathrm{Z}_{i,j,j+1}^{n,m,q}$	Placement of link between VNFs j and $j+1$ of SFC i on qth physical link between nodes n and m (Boolean)
$\mathcal{W}_i$	Allocated bandwidth per user for SFC i (Integer)
Request	
$\nabla_{r,i,j}$	Required amount of computing resource r for VNF j of SFC i
$\mathcal{U}_i$	Number of users in SFC i
ψ_i^-	Minimal required bandwidth value per user for SFC i
ψ_i^-	Maximal allowed bandwidth value per user for SFC i
$\mathcal{C}_{paid}$	Amount paid by the SFC owner to deploy the Service Chain
Network Infrastructure	
$\rho_{i,n,m}$	Number of physical paths between the nodes n and m that are allowed for SFC i
$\mathcal{R}_{r,n}$	Amount of computing resource r available on node n
$\zeta_{r,n}$	Cost per unit of resource r on node n
$\tau_l^{n,m,q}$	Link l is part of the qth path between nodes n and m (Boolean)
$\mathcal{R}_{\omega,l}$	Capacity of link l
ϕ_l	Latency of link l
ζ_l	Cost of using the link l per bandwidth unit
Physical Programming	
n_{sc}	Number of soft criteria.
g_i	Objective value for criterion i.
$\bar{g}_i$	Class function value for criterion i.
$\tilde{\bar{g}}^k$	Amplitude of interval k on y-axis.
λ_i^k	Amplitude of interval k on x-axis for the criterion i.
g_{is}	Upper/lower limit of interval k on x-axis for the criterion i (Class 1-S/2-S).
β	Convexity parameter

Table 4.1: Notations used

- **The bandwidth value is variable:** In that case $\mathcal{W}_i$ is an integer optimization variable, and we will denote by ψ_i^- the minimal amount of bandwidth per user that is required by the SLA of the SFC i, and by ψ_i^+ the maximum amount of bandwidth that is allowed for the SLA of the SFC i. Constraints 4.2 and 4.3 ensure that the allocated bandwidth per user for this SFC is within the allowed range of values for the selected SLA of the SFC i:

$$\mathcal{W}_i \leq \psi_i^+ \qquad \forall i \in \mathcal{S} \tag{4.2}$$

$$\mathcal{W}_i \geq \psi_i^- \qquad \forall i \in \mathcal{S} \tag{4.3}$$

On the other hand, Constraint 4.1 becomes quadratic as it features the product of a binary variable and an integer variable. It can be linearized using the big M method [161] as follows: we introduce the variable $\mathcal{H}_{i,j,j+1}^{n,m,q}$ which will take the value of the product of $\mathcal{W}_i$ and $\mathcal{Z}_{i,j,j+1}^{n,m,q}$. Constraint 4.1 then becomes: $\forall i \in \mathcal{S}, \forall l \in \mathcal{L}$:

$$\sum_{j \in \mathcal{V}_i} \sum_{n \in \mathcal{M}_{i,j}} \sum_{m \in \mathcal{M}_{i,j+1}} \sum_{q=1}^{\rho_{n,m}} \mathcal{H}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \cdot \mathcal{U}_i \leq \mathcal{R}_{\omega,l} \tag{4.4}$$

Constraints 4.5 to 4.8 ensure that the variable $\mathcal{H}_{i,j,j+1}^{n,m,q}$ effectively takes the value of the product of $\mathcal{W}_i$ and $\mathcal{Z}_{i,j,j+1}^{n,m,q}$. $\forall i \in \mathcal{S}, \forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_{i,j}, \forall m \in \mathcal{M}_{i,j+1}$:

$$\mathcal{H}_{i,j,j+1}^{n,m,q} \leq \psi_i^+ \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \tag{4.5}$$

$$\mathcal{H}_{i,j,j+1}^{n,m,q} \leq \mathcal{W}_i \tag{4.6}$$

$$\mathcal{H}_{i,j,j+1}^{n,m,q} \geq \mathcal{W}_i - (1 - \mathcal{Z}_{i,j,j+1}^{n,m,q}) \cdot \psi_i^+ \tag{4.7}$$

$$\mathcal{H}_{i,j,j+1}^{n,m,q} \geq 0 \tag{4.8}$$

Latency Constraint

Depending on the SLA of the SFC owner (Low Latency, Best Effort...), latency constraints may apply. As a reminder, in the previous chapter, the end-to-end latency for an SFC i was computed using the expression in 4.9, with ϕ_l being the link l's latency. Constraint 4.10 ensures that the end-to-end latency doesn't exceed the maximum allowed value for SFC i, which is denoted by ϕ_i^+.

$$\phi_i = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_{i,j}} \sum_{m \in \mathcal{M}_{i,j+1}} \sum_{q=1}^{\rho_{n,m}} \phi_l \cdot \mathcal{Z}_{i,j,j+1}^{n,m,q} \cdot \tau_l^{n,m,q} \tag{4.9}$$

$$\phi_i \leq \phi_i^+ \tag{4.10}$$

Cost Constraint

In order to ensure a profit margin to the SFC provider, the cost of deploying an SFC should not exceed a certain value. Similar to the previous contribution, computational and link costs for the SFC i respectively are computed as expressed in Equations 4.11 and 4.12, with $\zeta_{r,n}$ being the cost of using the resource r on node n per unit, and ζ_l the cost of using the link l per bandwidth unit. The overall cost $\mathcal{C}_i$ is the sum of the computational and link cost.

$$\mathcal{C}_{i,computing} = \sum_{n \in \mathcal{N}_d} \sum_{r \in \Re} \sum_{j \in \mathcal{V}_i} \zeta_{r,n} \cdot \nabla_{r,i,j} \cdot \mathcal{X}^{\mathrm{n}}_{i,j} \tag{4.11}$$

$$\mathcal{C}_{i,link} = \sum_{j \in \mathcal{V}_i} \sum_{l \in \mathcal{L}} \sum_{n \in \mathcal{M}_{i,j}} \sum_{m \in \mathcal{M}_{i,j+1}} \sum_{q=1}^{\rho_{n,m}} \zeta_l \cdot \mathcal{W}_i \cdot \mathcal{U}_i \cdot \mathcal{Z}^{n,m,q}_{i,j,j+1} \cdot \tau^{n,m,q}_l \tag{4.12}$$

$$\mathcal{C}_i = \mathcal{C}_{i,link} + \mathcal{C}_{i,computing} \tag{4.13}$$

We will denote by $\mathcal{C}_{i,paid}$ the amount paid by the SFC owner in order to have its Service Chain deployed, and by γ the minimal margin of profit that the operator requires. Constraint 4.14 ensures that the operator keeps its margin above the minimal value.

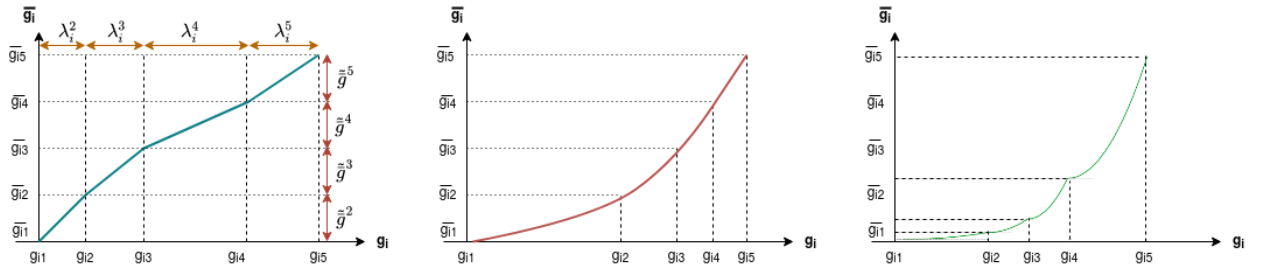
$$\mathcal{C}_i \leq (1 - \gamma) \cdot \mathcal{C}_{i,paid} \tag{4.14}$$

4.3.2 Objective Function

The objective of our model is to minimize the cost $\mathcal{C}_i$ and end-to-end latency ϕ_i, and maximize the allocated bandwidth per user $\mathcal{W}_i$. Since increasing the allocated bandwidth increases cost, and lower latency links are more expensive, the optimization's goal is to find a fair trade-off between all of these objectives while taking into account the tenant's preferences as expressed in the SLAs.

As stated earlier, we use Physical Programming in order to obtain the multi-criteria objective function. We use the variable i to designate a given criterion (in our case the criteria are the cost, latency and bandwidth), and the variable k to designate a class function interval or range. Figure 4.3 features examples of class functions $\bar{g}_i$ of the objective values g_i for all three methods. As illustrated in Figure 4.3a, we will denote by λ^k_i the x-axis amplitude of each criterion i in a given range k, and by $\tilde{\bar{g}}^k$ the y-axis amplitude for the range k. Note that the λ^k_i are given by the chosen SLA associated with the SFC to deploy. The class function value for each range limits is the same across criteria as expressed in relation 4.15.

$$\bar{g}_k \equiv \bar{g}_i(g_{i,k}) \forall i; (2 \leq k \leq 5); \bar{g}_1 \equiv 0 \tag{4.15}$$



(a) Linear Physical Programming (b) Non Linear Physical Programming (c) Global Physical Programming (n=2)

Figure 4.3: Examples of class functions $\bar{g_i}$ of the objective values g_i for the different Physical Programming methods

This means that the y-axis amplitude for each range $\tilde{\bar{g}}^k$ is also the same across criteria. Note that since the objective of our optimization is to reduce cost and latency, they are expressed using the Physical Programming class 1-S, and bandwidth per user is expressed using the class 2-S as the optimization aims to increase it.

Once the function for each criterion i has been defined, the global objective function $G(x)$ is formulated as follows:

$$\min G(x) = \log_{10} \left\{ \frac{1}{n_{sc}} \sum_{i=1}^{n_{sc}} \bar{g}_i[g_i] \right\} \tag{4.16}$$

The individual objective function $\bar{g}_i[g_i]$ for each criterion i is expressed differently depending on the chosen Physical Programming method as will be shown in the following.

Linear Physical Programming

This method has been proposed by Messac *et al.* in [151]. The piece-wise functions for each range are convex and linear as illustrated in Figure 4.3a.

Interval limits on the y-axis

$$\bar{g}_1 = 0, \quad \bar{g}_2 = \tilde{\bar{g}}^2 \text{ is a small positive number (e.g 0.1)} \tag{4.17}$$

In order to enforce the One Vs Others rule, the following relationship is applied:

$$\tilde{\bar{g}}^k = \beta(n_{sc} - 1)\tilde{\bar{g}}^{k-1}; (3 \leq k \leq 5); n_{sc} > 1; \beta > 1 \tag{4.18}$$

Where β is a convexity parameter that needs to be determined.

Convexity In a contribution from Ma *et al.* [162], a simplified method is proposed for obtaining the appropriate value of β for each criterion in one iteration using the following inequality system:

$$\begin{cases} \beta > 1 \\ \beta > \frac{g_{i,k}}{g_{i,k-1}(n_{sc}-1)}; \ (3 \leq k \leq 5) \end{cases}$$

Solving this system produces a value for β that satisfies the OVO-rule as well as the convexity requirement.

Class Function Taking the example of the class 1-S, the piece-wise linear function is expressed as follows:

$$\bar{g}_i = \begin{cases} \bar{g}_{k-1} + \tilde{\bar{g}}^k \left(\frac{g_i - g_{i,k-1}}{\lambda_i^k} \right) & \text{If } g_i \in [g_{i,k-1}, g_{i,k}], 2 \leq k \leq 5 \\ 0 & \text{If } g_i < g_{i,1} \end{cases} \tag{4.19}$$

Non Linear Physical Programming

This method has been detailed in [150]. The piece-wise functions are convex but non-linear, and can be arbitrarily shaped in order to reflect the priorities of the Decision Maker as illustrated in Figure 4.3b. This method is the most flexible compared to the LPP and GPP methods, and allows the most accurate depiction of the Decision Maker's preferences. However, it is also the most complex mathematically. In the following, we provide the mathematical formulation of the NLPP class functions, as well as the equation system that allows the computation of the convexity parameter β. More details on the method that was employed to obtain these equations can be found in the original NLPP paper.

Range limits on the y-axis

$$\bar{g}_1 = \tilde{\bar{g}}^1 \text{is a small positive number (e.g 0.1)} \tag{4.20}$$

$$\tilde{\bar{g}}^k \equiv \bar{g}_i[g_{i(k)}] - \bar{g}_i[g_{i(k-1)}] = \beta n_{sc} \tilde{\bar{g}}^{k-1}, 2 \leq k \leq 5 \tag{4.21}$$

Convexity The convexity parameter β must be greater than 1 and ensure that both constants a and b are strictly positive. β is first set at 1.5 then gradually increased by 0.5 until strictly positive values are obtained for the following a and b values for each range of each criterion:

$$a = \frac{3[3s_{i,k} + s_{i,k-1}] - 12\tilde{s}_i^k}{2(\lambda_i^k)^3} \tag{4.22}$$

$$b = \frac{12\tilde{s}_i^k - 3[s_{i,k} + 3s_{i,k-1}]}{2(\lambda_i^k)^3} \tag{4.23}$$

Where $\tilde{s}_i^k$ is a characteristic slope for the kth range of the criterion i as expressed in Eq 4.24. s_i^k allows the computation of the slope of a range according to the slope of the preceding range as is shown in Eqs 4.25-4.26.

$$\tilde{s}_i^k = \tilde{\bar{g}}^k / \lambda_i^k \tag{4.24}$$

$$s_{i1} = \alpha \cdot \tilde{s}_i^2, 0 < \alpha < 1 \tag{4.25}$$

$$s_{i,k} = \frac{(8\alpha+4)\tilde{s}_i^k - (8\alpha+1)s_{i,k-1}}{3}, 2 \leq k \leq 5 \tag{4.26}$$

Note that the constant α should be kept at a relatively small value (<0.1) in order to maximize the range of acceptable slopes at the range boundaries.

Class Functions **For each objective i:** Taking the example of the soft class 1-S, the following function is given:

– For the first range ($(g_i \leq g_{i,1})$)

$$\bar{g}_i = \bar{g}_1 \cdot e^{\left[(s_{i,1}/g_{i,1})(g_i - g_{i,1})\right]} \tag{4.27}$$

– For each other range

$$\bar{g}_i = T_0(\xi_i^k)\bar{g}_{k-1} + T_1(\xi_i^k)\bar{g}_k + \bar{T}_0(\xi_i^k; \lambda_i^k)s_{i,k-1} + \bar{T}_1(\xi_i^k; \lambda_i^k)s_{ik} \tag{4.28}$$

$$\xi_i^k = \frac{g_i - g_{i,k-1}}{\lambda_{i,k}} \tag{4.29}$$

Where $0 \leq \xi_i^k \leq 1, k = 2, ..., 5$, and

$$T_0(\xi) \equiv \tfrac{1}{2}\xi^4 - \tfrac{1}{2}(\xi - 1)^4 - 2\xi + \tfrac{3}{2} \tag{4.30}$$

$$T_1(\xi) \equiv -\tfrac{1}{2}\xi^4 + \tfrac{1}{2}(\xi - 1)^4 + 2\xi - \tfrac{1}{2} \tag{4.31}$$

$$\bar{T}_0(\xi, \lambda) \equiv \lambda[\tfrac{1}{8}\xi^4 - \tfrac{3}{8}(\xi - 1)^4 - \tfrac{1}{2}\xi + \tfrac{3}{8}] \tag{4.32}$$

$$\bar{T}_1(\xi, \lambda) \equiv \lambda[\tfrac{3}{8}\xi^4 - \tfrac{1}{8}(\xi - 1)^4 - \tfrac{1}{2}\xi + \tfrac{1}{8}] \tag{4.33}$$

Global Physical Programming

Global Physical Programming is an adaptation of the Physical Programming method proposed by Sanchis *et al.* [152], which allows a simpler formulation of the problem than the NLPP method, while still satisfying the previously detailed PP rules such as the OVO-rule. It also allows more flexibility in defining N, the number of preference ranges for each criterion.

Interval limits on the y-axis The images $\bar{g}^k$ at the range boundaries g_i^k are calculated as follows:

$$\bar{g}_0 = 0, \bar{g}_1 = \bar{g}^{ini}, \bar{g}^{ini} \geq 0 \tag{4.34}$$

$$\bar{g}^k = \bar{g}^{k-1} \cdot n_{sc} (2 \leq k \leq N) \tag{4.35}$$

With N being the number of preference ranges, and $\bar{g}^{ini}$ a pre-defined parameter.

Class Function The class function for the class 1-S is illustrated in Figure 4.3c and formulated as follows:

$$\bar{g}_i = \begin{cases} \bar{g}_{k-1} + \tilde{\bar{g}}^k \left(\frac{g_i - g_{i,k-1}}{\lambda_i^k} \right)^n & \text{If } g_i \in [g_{i,k-1}, g_{i,k}], 2 \leq k \leq 5 \\ 0 & \text{If } g_i < g_{i,1} \end{cases} \tag{4.36}$$

Where n is a pre-defined parameter.

4.4 Exact Solution

The problem as formulated is a set of piece-wise functions, and contains exponentiations that can reach the value of 4 in the case of Non-Linear Physical Programming. Therefore, the traditional ILP optimizers can't be used in order to solve the problem, as they are limited to quadratic models (polynomial with degree of 2). Although it is possible to implement the LPP method on one of the traditional solvers, such as CPLEX [163] or Gurobi [147], we preferred to use the same platform and configuration for all three methods for consistent results. In order to implement the exact resolution of the problem using the given formulation, we propose a Branch and Bound based algorithm that recursively constructs solutions while eliminating branches that don't satisfy the constraints, as well as those with a fitness (objective function value) that is worse than the one of the current best complete solution.

Topology

The algorithm takes as input the topology graph $\mathcal{G}$ which comprises the resource information of the nodes and links, the pre-computed set of paths $\mathcal{P}$, the request req which contains the resource requirements for each VNF, the number of users of the SFC, and its SLA class from which we can determine the latency and bandwidth constraints. The set of authorized nodes for placement $\mathcal{M}$ is also provided. The output of the algorithm is the optimal solution $bestSol$.

Solution Encoding

The solutions are encoded as a vector where the first value is $\mathcal{W}_i$ the allocated bandwidth per user for the SFC i, and each value at the index $2 * j + 1$ represents the ID of the node where the VNF j has been placed, while each value at the non-zero index $2 * j$ represents the index $\mathcal{P}^{n,m}$ of the path that will be used for the virtual link between the nodes where VNFs $j - 1$ and j have been mapped.

Branch and Bound Algorithm

Algorithm 3 features the pseudo-code of the proposed method. First, an initial valid solution is generated, and serves as the best solution reference in order to eliminate the partial solutions that are already worse. Then, as shown in Figure 4.4, the set of possible solutions is recursively constructed and evaluated level by level as follows: At the level $zero$, solutions using all of the possible bandwidth values are added, then for each subsequent level (i.e. VNF), solutions using the possible placement nodes for that VNF are created, while also computing the best path between the VNF and its predecessor. Once the partial solutions for each level are generated, if they satisfy the constraints, their fitness is calculated according to the SLA of the request and the Physical Programming method that is enforced. This fitness value is then compared to the fitness of the best solution found so far. If the fitness is worse, the solution is withdrawn and the branch is cut; otherwise, if the solution is complete, it becomes the best solution found so far. After all of the possible solutions for a level have been explored, the function *solGen* is recursively called again in order to compute the solutions of the next level. At the end of the algorithm (the last level is reached), the best solution is returned.

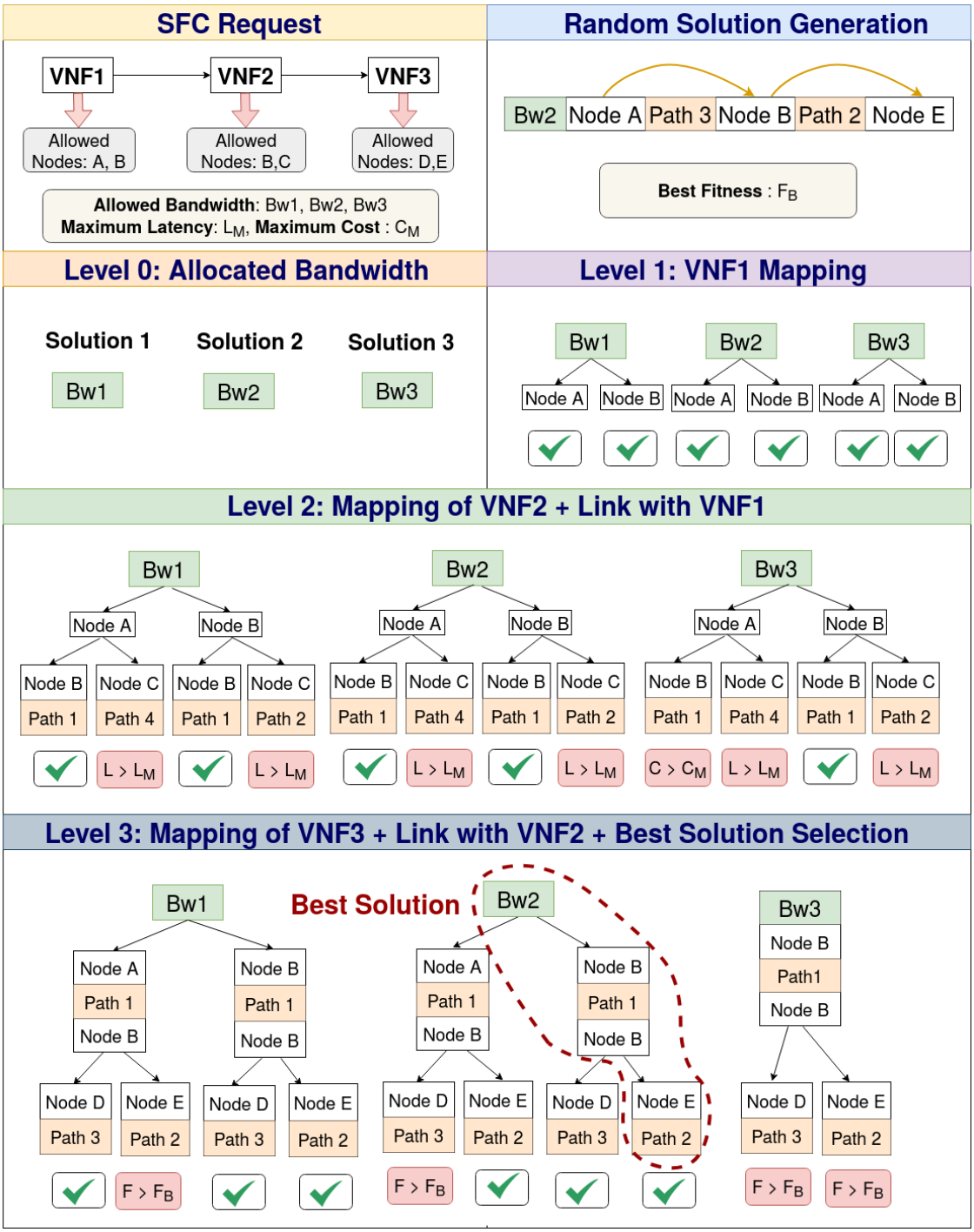


Figure 4.4: Branch and Bound Recursive Algorithm

Algorithm 3: Branch and Bound Placement Algorithm

Input : Topology $\mathcal{G}$, Paths $\mathcal{P}$, Request req
Set of Authorized Nodes $\mathcal{M}$
Output: Placement pl

Function *solGen($partS$,lvl,$\mathcal{M}$,req,$\mathcal{G}$,$\mathcal{P}$,$partC$,$partL$)* **:**
  $newPartS \leftarrow []$;
  for $sol \leftarrow 0$ **to** $|partS|$*-1* **do**
    for $j \in \mathcal{M}[lvl]$ **do**
      $currSol \leftarrow [i$ for $i \in partS[sol]]$;
      if *lvl= 0* **then**
        $currSol \leftarrow currSol + [j]$;
      else
        Compute $path$ the index of the best path between the last node in $currSol$ and j;
        $currSol \leftarrow currSol + [path, j]$;
      end
      if $currSol$ *satisfies the constraints* **then**
        Compute $currFit$ the fitness of solution $currSol$;
        if $currFit < bestFit$ **then**
          $newPartS \leftarrow nwPartS + [currSol]$;
          if $lvl = maxLvl$ **then**
            $currBestSol \leftarrow currSol$;
            $bestFit \leftarrow currFit$;
          end
        end
      end
    end
  end
  if $lvl = maxLvl$ **then**
    return $currBestSol$
  else
    solGen($newPartS$,lvl+1,$\mathcal{M}$,req,$\mathcal{G}$,$\mathcal{P}$)
  end
end
Generate one initial solution $currBestSol$ that satisfies the constraints, and save its fitness in $currFit$;
Determine the set of allowed bandwidth values $\mathcal{B}$ from the SLA of the request;
$partSol \leftarrow [[b]$ for $b \in \mathcal{B}]$;
$lvl \leftarrow$0;
$bestSol \leftarrow solGen$(partS,lvl,M,req,G,P)
return *bestSol*

4.5 Heuristic Solution

Similar to the previous chapter, we develop a memetic algorithm that implements the different constraints that were formulated in Section 4.3. For a number of generations N, new placement combinations are explored using solution generation, mutation, crossover, or local improvement operations, the obtained solutions are then evaluated, and the best ones are retained for the next iteration of the heuristic. However, in this contribution, we employ the Physical Programming objective function, in order to evaluate our solution, thus optimizing the relative cost, the end-to-end latency, and the bandwidth per user.

4.6 Evaluation

4.6.1 Test Environment

Our simulations were conducted on a server with 32 Intel Xeon 2.60 GHz CPU cores, 128 GB of memory, hosting an Ubuntu Server 16.04 x64 OS. The evaluation scripts, as well as the Branch and Bound and memetic algorithms were developed using the Python programming language, and the Gurobi solver [147] was used in order to implement the ILP model.

4.6.2 Topology and Requests

In order to evaluate our solutions, we leveraged on the *networkx* [148] library, which allows generating multi-domain topologies where each domain is a 3-level Fat Tree network, which is a network topology that is used in data-centers [164]. The physical servers are connected to edge switches that are connected to aggregate switches, and are in turn connected to core switches that ensure the inter-domain communication. We also use this library in order to compute the set of paths between nodes.

The generated multi-domain topologies are of three types, as depicted in Table 4.2. For the link characteristics, we follow the fat tree concept by allocating more bandwidth to the links as we go higher in the topology [165]. In terms of latency, two types of links are used: low latency links, and less expensive high latency links. The servers that host VNFs can be of 3 types with different computing resource capacities, the hardware configurations of each server and link type are detailed in Table 4.2. Note that these types for each component are randomly selected with equal probabilities.

Topology Configurations			
Topology	**Domain Number**	**Node Number**	**Link Number**
T1	4	144	240
T2	6	216	368
T3	8	288	496
Link Configurations			
Link Type	**Bw (Gbps)**	**Latency (ms)**	**Price per unit**
Server-Edge	10	0.1-0.5	10-20
Edge-Aggregation	10	0.5-1	40-80
Aggregation-Core (Low Latency)	40	0.5-1	480-960
Aggregation-Core (High Latency)	40	10-20	160-320
Inter-domain (Low Latency)	100	1-5	1920-3840
Inter-domain (Medium Latency)	100	10-15	1280-2560
Inter-domain (High Latency)	100	25-50	640-1280
Node Configurations			
Node Type	**CPU**	**RAM (Gb)**	**Disk (To)**
Large	500	1024	256
Medium	250	256	64
Small	100	64	16
Cost Per unit	10-20	10-20	1-2
VNF Resource Requirements			
VNF Type	**CPU**	**RAM (Gb)**	**Disk (Go)**
Large	4	6	80
Medium	2	4	40
Small	1	2	20

Table 4.2: Topology and Request Characteristics

As for the SFC requests, they are randomly generated with VNF lengths of 4 to 10. Each VNF can be of one of three types (Large, Medium, Small) with different resource requirements as detailed in Table 4.2. Each SFC serves as a slice that accommodates a certain number of users, and we randomly set the maximum number of users for each SFC in the interval *10-5000*.

In order to support the 5G use cases, we also set different SLA types to each request based on the 5G service types (URLLC, mMTC, eMBB), and different SDO documents [166–169]. We can identify many SLA classes with different requirements regarding the QoS metrics:

- **Low Latency - High Throughput:** Such as Augmented/Virtual Reality applications, live video streaming with strict latency and bandwidth requirements.
- **Low Latency - Low Throughput:** For mission-critical signaling in scenarios such as Autonomous Driving (V2X messages) and Smart Factories, with strict latency requirements.
- **Medium Latency - High Throughput:** For buffered video streaming and general file sharing applications.
- **Medium Latency - Low Throughput:** For other TCP-based services such as web browsing, messaging etc.

Additionally, in the general case of classic internet usage, operators provide different offers of level of service to their customers that vary in cost (Platinum, Gold, Silver, Best Effort). In this vein, we set 6 different SLA classes with strict and non-strict QoS requirements, as described in Table 4.3.

SLA Class	Use Case	Latency	Bandwidth
0	Real-time video applications (AR,VR,live streaming)	Strict: $<$ 10ms	Guaranteed: $>$ 1 Gbps
1	Mission-Critical Signaling (V2X, Smart Factories)	Strict: $<$ 10ms	Guaranteed: $>$ 10 Mbps
2	Internet Usage, Platinum	$<$ 100ms	$>$ 100 Mbps
3	Internet Usage, Gold	$<$ 200ms	$>$ 50 Mbps
4	Internet Usage, Silver	$<$ 300ms	$>$ 25 Mbps
5	Internet Usage, Best Effort	Best Effort	Best Effort

Table 4.3: SLA Classes

These requirements will next be translated to Physical Programming classes for use in the multi-objective optimization scheme:

- In the case of SLA classes 0 and 1, latency and bandwidth requirements are strict and are therefore equivalent to Physical Programming hard classes 1-H (Must be smaller) and 2-H (Must be greater) respectively. They can be modeled as linear constraints.
- For SLA classes 2 to 4, latency and bandwidth requirements are less strict; they can be translated to Physical Programming soft classes 1-S (Smaller is better) and 2-S (Greater is better), respectively.

- SLA Class 5 is best effort; hence only the cost is minimized.
- The cost objective for all of the classes will need to be minimized and is equivalent to the Physical Programming soft class 1-S (Smaller is better).

Given these values, and considering that a tenant with a certain SLA shouldn't be able to benefit from a service level that corresponds to a higher nor to a lower level than its own, we can define the intervals of allowed values for the latency and bandwidth. However, the deployment cost always has to be inferior to 95% of the amount paid by the SFC owner irrespective of the SLA class, in order to guarantee a profit margin of at least 5% to the operator. Once the allowed intervals have been defined, we can determine the values of preference ranges for each objective and each SLA class. Table 4.4 details the intervals for the Physical Programming ranges that will be used in order to generate the evaluation functions of each objective. We denote by - the *Unacceptable* SLA range, by H-U the *Highly Undesirable* SLA range, by U the *Undesirable* range, by T the *Tolerable* range and by D and H-D the *Desirable* and *Highly Desirable* ranges respectively.

SLA Class	**-**	**H-Undesirable**	**Undesirable**	**Tolerable**	**Desirable**	**H-Desirable**
Latency (ms)						
2	>105	95-105	90-95	85-90	75-85	<75
3	>210	190-210	180-190 ms	170-180	170-150	<150
4	>315	285-315	270-285	255-270	225-255	<225
Bandwidth per User (Mbps)						
2	<95	95-105	105-115	115-125	125-135	>135
3	<47.5	47.5-52.5	52.5-57.5	57.5-62.5	62.5-67.5	>67.5
4	<23.75	23.75-26.25	26.25-28.75	28.75-31.25	31.25-33.75	>33.75
Relative Deployment Cost (%)						
All SLAs	>95	85-95	70-85	60-70	50-60	<50

Table 4.4: Physical Programming Ranges

Based on these SLA classes, we also set the deployment cost per SFC and per user for the SFC tenant as illustrated in Table 4.5:

SLA Class	0	1	2	3	4	5
Cost per User	3500	3000	2750	2500	2000	1500

Table 4.5: SLA deployment cost per SLA Class

Additionally, convexity parameters also need to be defined for each Physical Programming method. For NLPP, we set the constant α at 0.05, and the convexity parameter β at 1.5. In LPP, β is set at 1.1, and in GPP, α_{init} is set at 0.1 and the exponent n is set at 2. By applying these values, we obtain the objective functions for each objective and each SLA. Figure 4.5 illustrates the obtained functions for SLA class 2 for latency, bandwidth, and cost, respectively. For each figure, the blue line represents the Linear Physical Programming function, the red line represents the Non-Linear Physical Programming function, and the green line represents the Global Physical Programming function. It can be noticed that the objective function results for each objective at the preference ranges limits are equal, which normalizes the objectives.

4.6.3 Algorithms

In our simulations, we evaluate 3 different optimization algorithms:

- The heuristic algorithm combined with the SFC hierarchical partitioning framework and applied to both levels of placement with the three Physical Programming methods, and with a limited visibility on the domains at the first step (since the algorithm performs the initial placement on an abstracted view of the topology). Note that the number of generations N for which the heuristic is run is varied, in order to study its effect on the efficiency of the algorithm, depending on the SFC length and topology size. We use the following values: $N_1 = |req| * |\mathcal{G}|$, $N_2 = \frac{N_1}{4}$, $N_3 = \frac{N_1}{16}$. Where $|req|$ is the length of the request, and $|\mathcal{G}|$ is the size of the topology. Furthermore, since each local domain only receives part of the original SFC, the cost and latency limits for each sub-SFC are recalculated according to the sub-SFC length's proportion to the original SFC's length.

- In order to evaluate the efficiency of the heuristic algorithm we also implement the Branch and Bound algorithm applied on the full topology with the three Physical Programming methods.

- In order to evaluate the efficiency of Physical Programming compared to weighted sum MOO methods, we implement an ILP applied on the full topology using constraints expressed in Section 4.3, but where the objective function is a weighted sum of the normalized objectives:

$$\textbf{minimize} \qquad \alpha_{\mathcal{C}} \cdot \frac{\mathcal{C}_s}{\lambda_{\mathcal{C}}} + \alpha_{\phi} \cdot \frac{\phi}{\lambda_{\phi}} - \alpha_{\mathcal{W}} \cdot \frac{\mathcal{W}_i}{\lambda_{\mathcal{W}}}$$

  With $\alpha_{\mathcal{C}}$, α_{ϕ}, and $\alpha_{\mathcal{W}}$ being the associated weights to cost, latency, and bandwidth respectively, which are determined from the tenant's SLA as shown in Table 4.6. For SLA classes

0 and 1, the latency and bandwidth requirements are strict, which means that they are expressed as constraints. Their associated weights are therefore set to 0, and only cost is minimized in the objective function. The SLA classes 2 to 4 represent different levels of service for internet usage, where depending on the subscription, the operator would give more or less priority to the QoS related objectives (allocated bandwidth and latency) over the cost objective. The last SLA class is best effort, which means that only cost is considered, and the weights associated to latency and bandwidth are set to 0. We also denote by $\lambda_{\mathcal{C}}$, λ_{ϕ}, and $\lambda_{\mathcal{W}}$ the normalization coefficients for cost, latency, and bandwidth which are set as the optimal values for each objective for the selected SLA. The ILP is also applied on the complete multi-domain topology.

SLA	Cost $\alpha_{\mathcal{C}}$	Latency α_{ϕ}	Bandwidth $\alpha_{\mathcal{W}}$
0-1 (Strict QoS reqs.)	1	0	0
2 (Platinum)	0.2	0.4	0.4
3 (Gold)	0.5	0.25	0.25
4 (Silver)	0.8	0.1	0.1
5 (Best Effort)	1	0	0

Table 4.6: Weights associated to the optimization objectives for the ILP depending on the SLA

For the three algorithms, 150 requests of different SLAs are generated and placed sequentially, and after each SFC placement, the capacity values of the topology are updated. Note that since the output of the heuristic can vary from an experiment to another, we repeat the experiment 20 times.

4.6.4 Results

In the following, we present the results obtained from the different simulations. We evaluate the performance of our algorithms using two metrics:

- Efficiency, which can be quantified by classifying each solution in the corresponding SLA satisfaction class according to the obtained results, and comparing the overall results. We also evaluate the relative difference between the obtained results and the optimal value for each optimization objective. Note that for the cost objective, we evaluate the relative deployment cost, which is the ratio between the deployment cost, and the amount paid by the tenant in order to deploy its SFC.

- Scalability, which can be estimated by measuring the evolution of runtime for each algorithm when increasing SFC length and topology size.

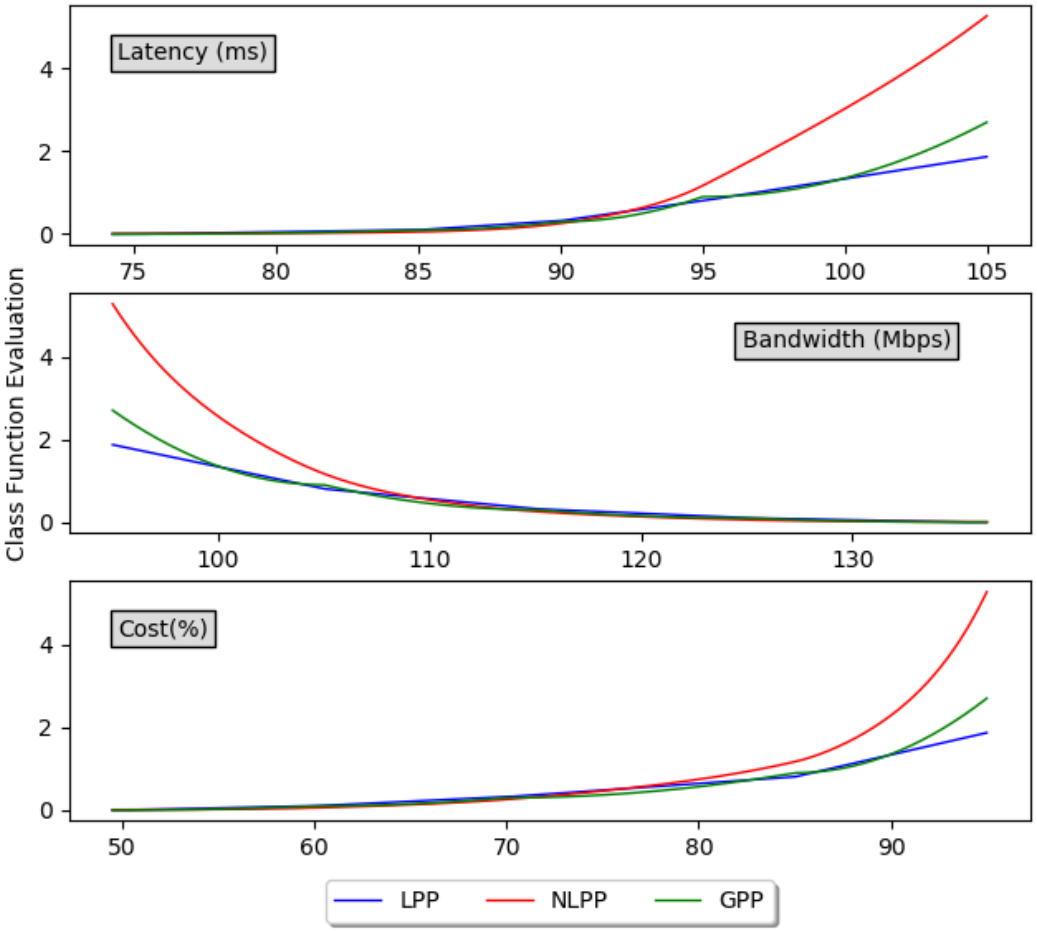


Figure 4.5: Physical Programming class functions for the SLA class 2

Efficiency

Overall Satisfaction Rate First, we evaluate the efficiency of the Physical Programming method compared to the ILP by classifying the placed solutions according to the preference ranges provided by the SFC owner. Table 4.7 provides the classifications of all of the requests regardless of

SFC length and topology type. It can be observed that while the ILP provides the best placement solutions in terms of latency with 100% of the solutions in the *Highly Desirable* range, it falls short in terms of allocated bandwidth per user where the minimum value is allocated in 100% of the time. For the relative deployment cost, 30% of the solutions are classified at the *Highly Undesirable* range. Only 65% of the request placements are classified at the *Highly Desirable* range, even when higher weights are assigned to the cost and bandwidth objectives. In comparison, the Physical Programming results maintain relatively good results across all objectives. Indeed, the Branch and Bound algorithm combined with Physical Programming provides solutions where the latency is in the *Highly Desirable* range 91 to 94% of the times, the cost is always kept at a minimal value, and the optimal value for bandwidth is reached for 95% to 97% of the requests. The reason for this difference between the results of the Physical Programming methods and the ILP lies in the fact that in Physical Programming, if a solution reaches the optimal value for an objective, the evaluation function keeps outputting the same value even if the objective is further optimized. Thus, giving preference to solutions that also improve the other objectives. However, the weighted-sum ILP might give preference to solutions that minimize a certain objective past its ideal value, while disregarding the other objectives.

Physical Programming methods It can also be noticed that among the Physical Programming methods, the Linear method is the most efficient. Although all of the methods perform similarly in terms of cost, the LPP method keeps the number of solutions in the *Highly Undesirable* range to a minimum with 0.58% of solutions for the latency and 1.08% of solutions for bandwidth, as opposed to 1.47% and 4.07% respectively for NLPP, and 1.59% and 2.8% of solutions for GPP respectively. The number of *Highly Desirable* solutions for LPP is also the highest with 94.99% of solutions as opposed to 91.31% and 91.62% of solutions for NLPP and GPP respectively regarding the latency, and 97.15% of solutions as opposed to 95.69% and 95.81% for NLPP and GPP respectively in terms of bandwidth. However, this difference remains in the range of $2 - 3\%$ of the produced solutions, which isn't a significant difference.

The Heuristic Since the LPP method performs better than the other two, and for the sake of brevity, we will only display the results of the heuristic using LPP, while varying the generation number. It can be noticed that the solutions from the heuristic are less optimal than those obtained using the Branch and Bound algorithm, with more solutions that are classified on the least desirable ranges. However, the heuristic still outperforms the ILP with results that remain at the *Highly Desirable* range for $93 - 96\%$ of the placement results. The change in the number of generations does not significantly affect on the overall results, as the results classification remains consistent with a slight decrease in the number of solutions classified in the *Highly Desirable* range, as the

number of generations decreases. Similarly to the Branch and Bound algorithm applied to the Physical Programming method, all objectives are uniformly optimized and no objective is given preference over the two others.

SLA Class	H-Undesirable	Undesirable	Tolerable	Desirable	H-Desirable
Latency					
B&B - LPP	0.58%	0.96%	2.29%	1.21%	94.99%
B&B - NLPP	1.47%	2.10%	2.54%	2.61%	91.31%
B&B - GPP	1.59%	2.04%	2.61%	2.16%	91.62%
ILP	0%	0%	0%	0%	100%
Heuristic - N1	1.15%	0.77%	0.85%	0.58%	96.68%
Heuristic - N2	1.16%	0.88%	0.98%	0.55%	96.46%
Heuristic - N3	1.05%	0.78%	0.85%	0.46%	96.89%
Bandwidth per User					
B&B - LPP	1.08%	0%	1.78%	0%	97.15%
B&B - NLPP	4.07%	0%	0.26%	0%	95.69%
B&B - GPP	2.8%	0%	1.40%	0%	95.81%
ILP	100%	0%	0%	0%	0%
Heuristic - N1	0.70%	0.38%	0.43%	0.52%	98%
Heuristic - N2	0.67%	0.46%	0.61%	0.52%	97.77%
Heuristic - N3	0.70%	0.41%	0.60%	0.58%	97.74%
Relative Deployment Cost					
B&B - LPP	0%	0%	0%	0%	100%
B&B - NLPP	0%	0%	0%	0%	100%
B&B - GPP	0%	0%	0%	0%	100%
ILP	29.97%	2.07%	1.57%	1.76%	64.64%
Heuristic - N1	0.30%	0.42%	0.29%	0.41%	98.61%
Heuristic - N2	0.36%	0.43%	0.36%	0.47%	98.40%
Heuristic - N3	0.23%	0.41%	0.31%	0.44%	98.63%

Table 4.7: Results Classification using Preference Classes

Effect of the SFC Length and Topology Size Next, we study how the efficiency of the 3 solutions is affected by the changes of SFC length, as well as topology size. Figures 4.6, 4.7 and 4.8 illustrate the obtained results for latency, bandwidth, and cost, respectively. The 0 value on the

y-axis represents the optimal value for each request, which is the *Highly Desirable* value for that SLA class, and the different results represent the relative mean difference between the obtained value for each algorithm and the optimal one. Depending on the optimization objective, it would be preferable to have results either above or below the optimal value threshold. Indeed, since the optimization model aims to reduce the end-to-end latency and relative cost, the best solutions should be below the threshold. In contrast, as the bandwidth per user is a Larger-is-Better Physical Programming class, the best solutions are above the threshold. The blue line represents the average ILP results, the orange, green and red lines represent the mean Branch and Bound results for the LPP, NLPP, and GPP methods respectively, and the purple, brown and pink dashed lines represent the mean and 95% Confidence Interval (C.I) of heuristic results using LPP for the generation numbers N1, N2, and N3 respectively.

Latency Looking at the end-to-end latency results in Figure 4.6, we can first confirm the results in Table 4.7, where the ILP gives preference to latency over the other objectives. The latency values for the ILP are under the optimal threshold with a mean value of around $95 - 100\%$ less than the objective value across SFC lengths and topology sizes. The same behavior can be observed from the values obtained from the Physical Programming algorithms regardless of the topology size and SFC length. As for the Branch and Bound and heuristic results, we can observe that latency values slowly increase proportionally with SFC length and topology size while still remaining under the optimal value threshold, and therefore within the *Highly Desirable* preference class with values increasing from -81.5% to -38.4% for the topology T1, -80.72% to -29.04% for the topology T2, and -68.75% to -20.33% for the topology T3. It can also be noticed that although all of the Physical Programming plots are similar, the NLPP latency results are slightly lower than those of the other two methods for the Branch and Bound algorithm for the shorter SFCs with values of -83.63%,-80.72%, and -68.75% for SFC length of 4 SFCs and topologies T1,T2, and T3 respectively. However, as SFC length increases, the LPP method provides solutions with slightly lower latencies with values of up to -45.45%,-35%, and -32.81% for SFC length 10 and topologies T1,T2, and T3 respectively. Another observation that can be made on the heuristic results is that the highest generation numbers N1 and N2, provide, on average, lower latencies than those obtained with the generation number N3. This can be explained by the fact that the longer a heuristic is run, the more solutions are explored and the better the results are.

Bandwidth In contrast, the ILP results for bandwidth (Figure 4.7) remain stable and under the optimal value threshold for all SFC lengths and all topologies with mean values of around -25%. Note that in Table 4.7 all of the placement solutions are classified as *Highly Undesirable*. As for the Branch and Bound results, they also remain stable but slightly over the optimal value threshold,

which is consistent with the results in Table 4.7 where the majority of solutions are classified as *Highly Desirable*. The NLPP method provides the highest bandwidth values with an average of values that are 8% higher than the optimal value for lower SFC lengths, and slowly decreasing to 0.9% as the number of VNFs increases, as opposed to $6 - 2\%$ for LPP and $7 - 0.1\%$ for GPP.

However, the bandwidth values for the heuristic algorithm fluctuate more depending on SFC length and topology size. Indeed, for topology T1, the mean values of bandwidth per user for the heuristic algorithm are higher than the optimal value and those of the Branch and Bound solution. The values oscillate between 0.04% and 79% over the *Highly Desirable* threshold, with C.Is of $0.4 - 21\%$ respectively, with a general tendency to decrease as the SFC length increases. The heuristic results for bandwidth stabilize as topology size increases. For topology T2, the mean values of the bandwidth per user are reduced, but are still close to the optimal value with results that are between 6% below the threshold, and 23% over the threshold, and C.Is of $0.4 - 12\%$. As for topology T3, the mean bandwidth values are very similar to those of the Branch and Bound algorithm for the shorter SFCs with values of around $2 - 4\%$, and a peak at SFC lengths 9 and 10 that reaches 40% over the optimal value with a C.I of 11% for generation number N3; while the mean values for generation numbers N1 and N2 reach the values 20% and 24% over the threshold respectively for the SFC length of 10 VNFs, with C.Is of 8% and 9% respectively.

Relative Cost Moving on to the relative cost, we can see in Figure 4.8 that once again the Branch and Bound results remain stable and way under the optimal value across configurations with mean values of -96% to -80%. We can also observe that for this objective, the GPP method slightly outperforms the LPP and NLPP methods. The heuristic algorithm also displays values that slowly increase as SFC length increases, but the results remain similar to those obtained by the Branch and Bound algorithm. In contrast, as the ILP provides lower latency values, hence using higher-cost links compared to the other solutions, its cost results are higher than those of the algorithms that leverage on Physical Programming. For topology T1 the relative cost increases proportionally to the SFC length with a mean relative cost of -82% to -49%; while for topology T2 the ILP provides results with relative costs of 79% to 49% lower than the optimal value. For topology T3 the relative cost oscillates between -75% and -60%.

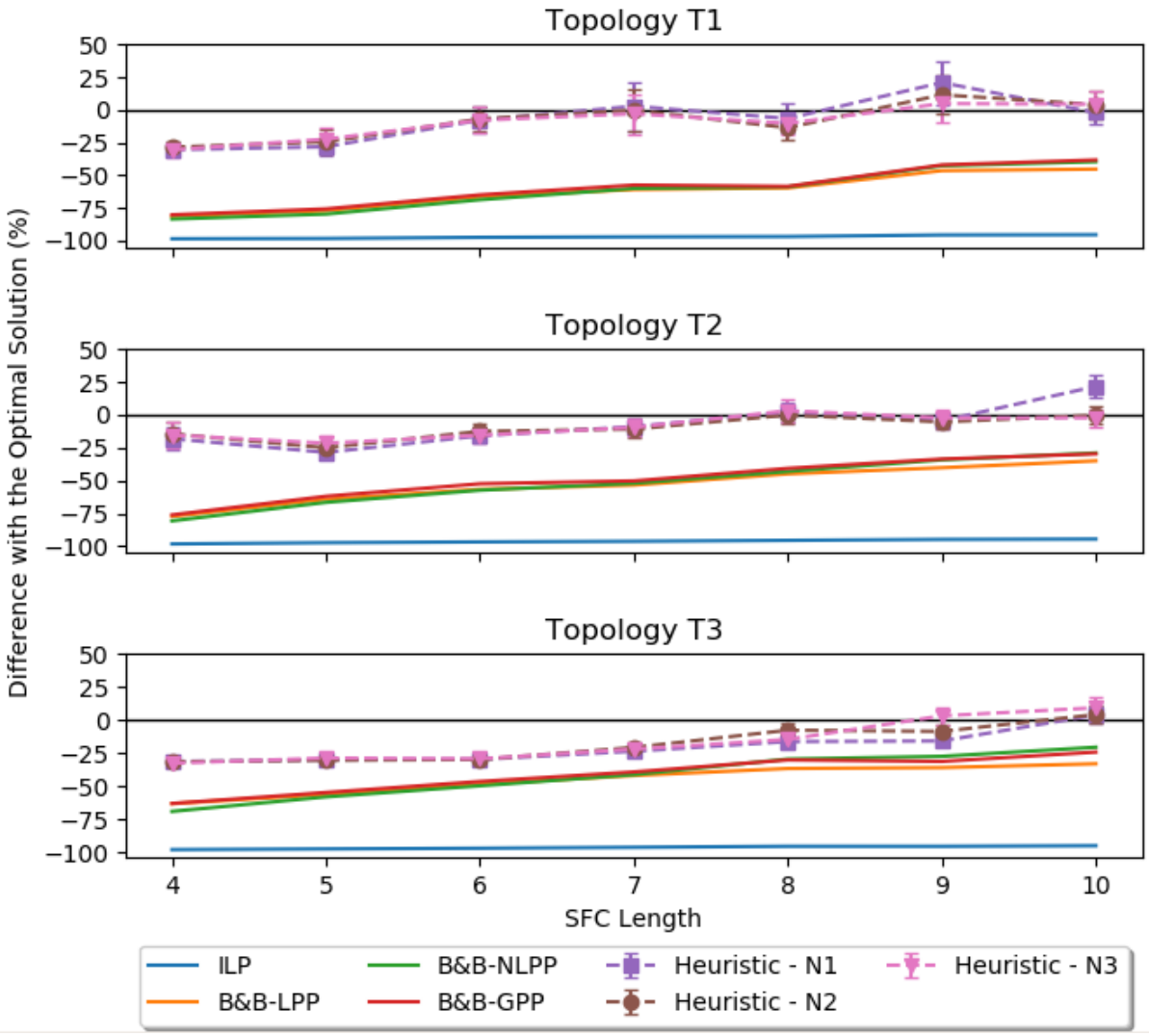


Figure 4.6: Relative Difference Between the Placement Solutions and the Optimal Value for the End-to-End Latency

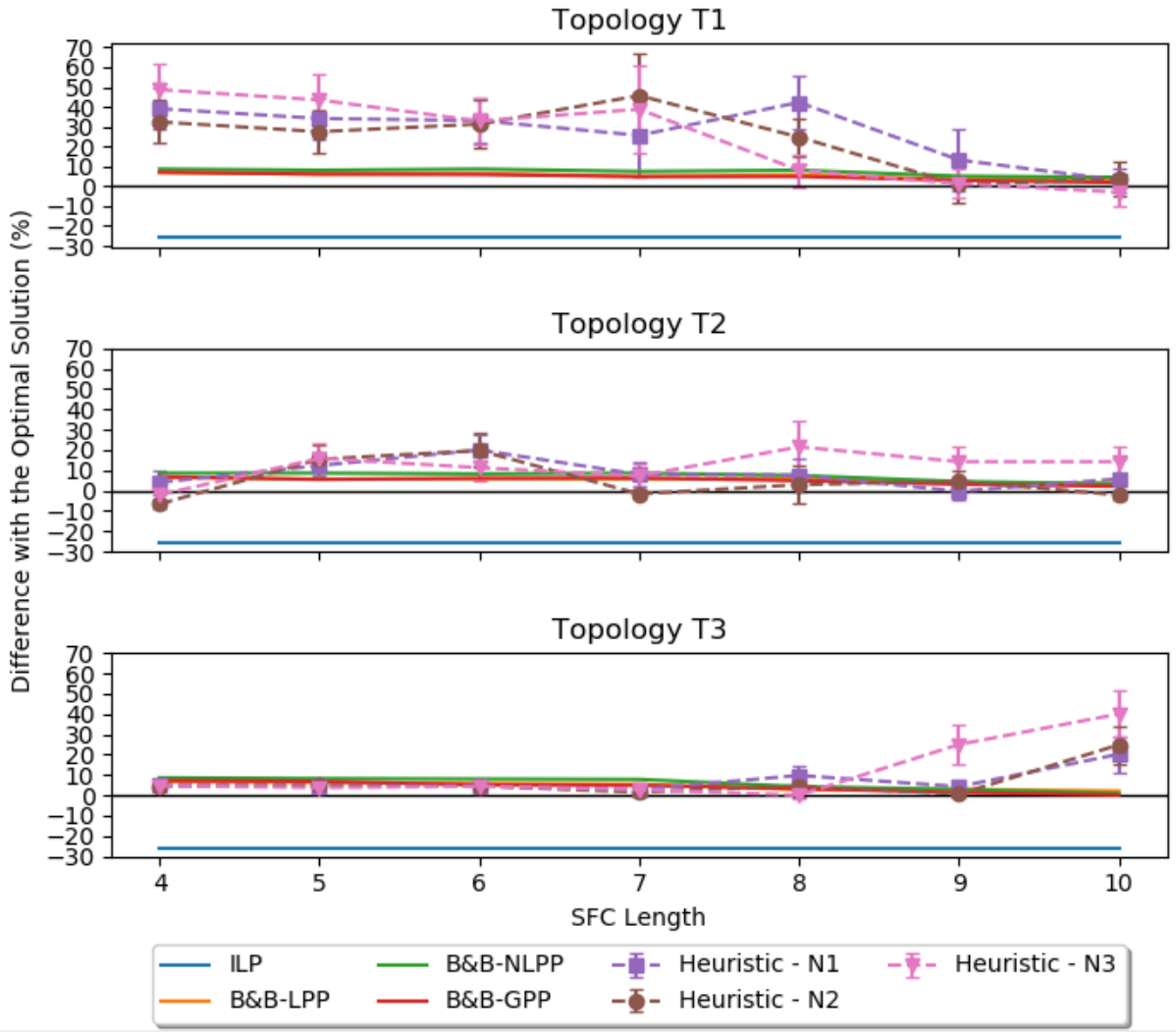


Figure 4.7: Relative Difference Between the Placement Solutions and the Optimal Value for the Bandwidth per User

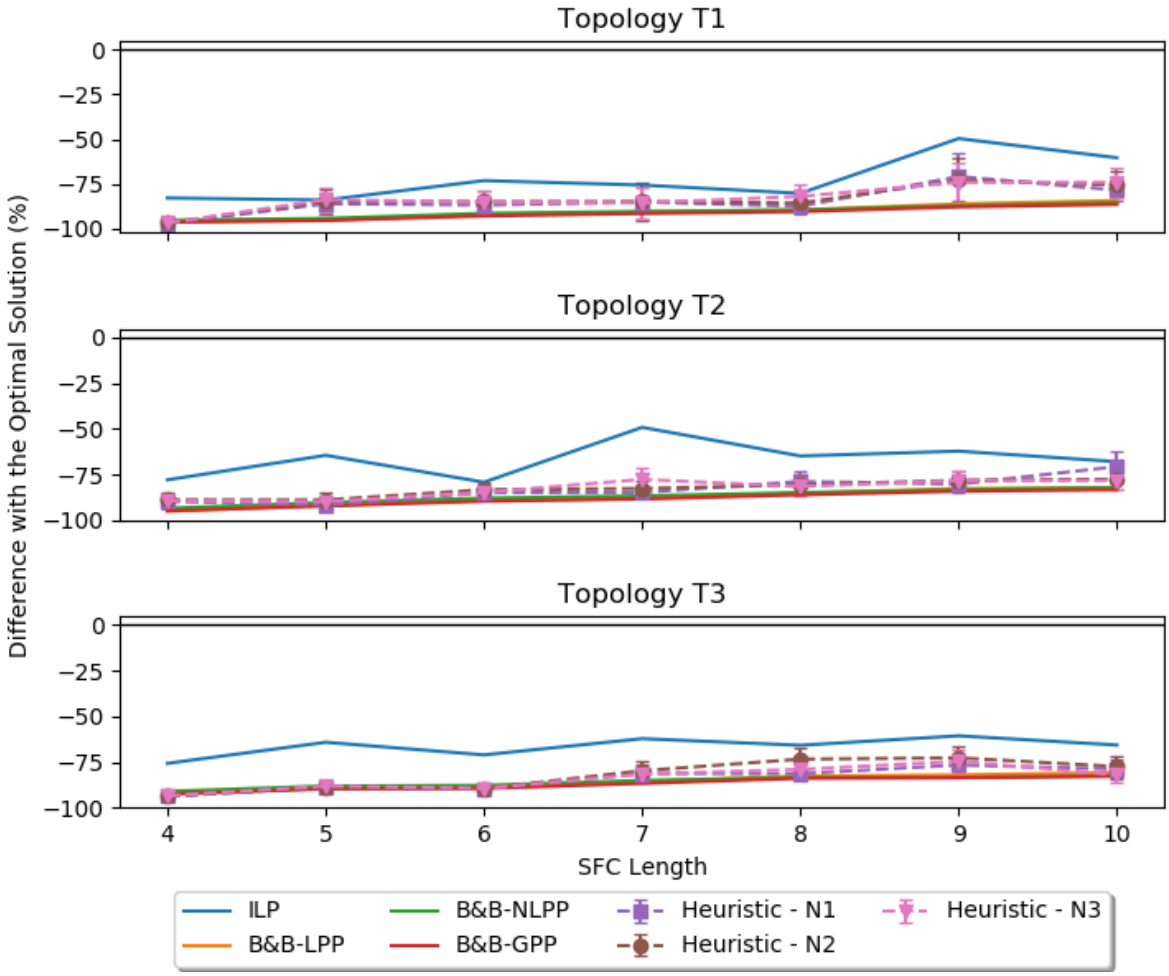


Figure 4.8: Relative Difference Between the Placement Solutions and the Optimal Value for the Relative Cost

Runtime

During our simulations, we also measured the mean runtime of the algorithms in order to evaluate their scalability in terms of computing time. The results are gathered in Table 4.8. Note that depending on the quality of the first generated solution, the execution times of the Branch and Bound algorithm might vary. Indeed, a relatively good first solution might allow the elimination of mul-

tiple branches at the first levels, therefore significantly reducing the solution space to explore. In contrast, the heuristic runtimes are stable, as they depend essentially on the number of generations that has been set as an input of the algorithm.

Exact Solutions Due to the complexity of the exact formulation, the ILP displays high computation times for all of the simulation configurations. For topology T1, the runtime for the ILP starts at 19 seconds for SFC length of 4 and increases to up to 122 seconds for SFC length 10. Similar behavior is observed for topologies T2 and T3 with an increase from $24.58s$ and $24.19s$ for SFC length 4 to $110s$ and $102s$ for SFC length 10 respectively.

The Branch and Bound algorithm turns out to be more costly than the ILP in terms of runtime; indeed, the runtime for longer SFC lengths increases to up to $249s$. Overall, the NLPP method is the one with the highest runtime values starting at 25.95 seconds for SFC length of 4 and increasing to 186 seconds for SFC length 10 when used on topology T1, when for topologies T2 and T3 with it increases from 24.58 and 24.19 for SFC length 4 to $110s$ and $102s$ for SFC length 10 respectively. The LPP method comes second with a runtime from $21s$ to $174s$ for topology T1, and $24s$ and $42s$ to $249s$ and $234s$ respectively for topologies T2 and T3. Lastly, Global Physical Programming displays runtimes from $20s$, $24s$, and $45s$ to $144s$, $205s$, and $209s$ for topologies T1, T2, and T3, respectively.

Heuristics In contrast, the heuristic proves the most scalable as it outputs results in significantly lower times compared to the exact solutions. The heuristic algorithm with the highest generation number N1 outputs results after $2.1s$ and up to $11.05s$ depending on the SFc size for the topology T1, while for the topologies T2 and T3 the proposed heuristic returns results after $2.89s$ to $11.65s$ and $8.09s$ to $19.46s$ respectively. Reducing the number of generations to N2 further reduces the runtime, with values of $1.06s$ to $3.5s$ for topology T1, and between $1.21s$ and $5.36s$ for topology T2, while for topology T3 the runtime ranges between $2.24s$ and $6.29s$. The generation number N3 produces the lowest runtimes with $0.71s$ to $1.64s$, $0.65s$ to $2.38s$, and $1.73s$ to $2.24s$ for topologies T1, T2, and T3 respectively. We can therefore conclude that the proposed exact solutions are not scalable, and are not suitable for use at runtime, while the heuristic approach allows results within less than 20 seconds for the biggest instances, and less than a second for the small instances when reducing the number of generations.

SFC Length	4	5	6	7	8	9	10
Topology T1							
ILP	19.36	30.54	44.16	55.13	78.76	82.94	122.26
B&B - LPP	21.02	25.23	47.8	65.67	97.69	149.04	174.67
B&B - NLPP	25.95	33.89	64.5	86.48	118.65	173.86	186.96
B&B - GPP	20.73	25.74	47.82	62.19	86.6	124.81	144.41
Heuristic - N1	2.87	2.1	4.2	6.21	6.66	9.11	11.05
Heuristic - N2	1.27	1.06	1.76	1.52	2.81	2.75	3.5
Heuristic - N3	0.86	0.71	0.93	0.91	0.95	1.41	1.64
Topology T2							
ILP	24.58	33.44	45.97	60.24	72.53	98.24	110.6
B&B - LPP	24.28	51.58	82.21	123.2	146.36	151.73	249.4
B&B - NLPP	27.34	63.15	95.22	129.86	146.12	154.96	218.9
B&B - GPP	24.22	51.74	79.21	112.72	129.12	137.25	205.17
Heuristic - N1	2.89	4.49	7.31	6.79	7.54	11.65	11.57
Heuristic - N2	1.21	2.64	2.83	2.01	3.29	5.36	4.22
Heuristic - N3	0.65	0.95	1.54	1.32	1.86	2.38	2.38
Topology T3							
ILP	24.19	35.29	45.13	56.22	77.96	102.43	102.95
B&B - LPP	42.49	60.92	98.14	143.99	171.21	232.39	234.92
B&B - NLPP	49.61	74.82	118.66	164.38	195.13	241.01	236.18
B&B - GPP	45.2	65.68	102.73	140.63	172.14	210.5	209.1
Heuristic - N1	8.09	10.84	13.43	12.4	19.46	16.93	15.84
Heuristic - N2	2.24	3.83	5.12	4.58	6.29	3.66	3.83
Heuristic - N3	1.73	1.86	1.61	1.85	3.15	2.74	2.24

Table 4.8: Comparison of computation times (in seconds)

4.6.5 Results Discussion

Comparison between Physical Programming and ILP

Thanks to our extensive evaluations with different simulation configurations, we are able to assess the efficiency of the Physical Programming method compared to the weighted-sum ILP. Indeed, simulation results demonstrate how our proposed method equally optimizes all three objectives and provides solutions that best reflect the DM's preferences. The results also indicate that the

Branch and Bound algorithm is relatively scalable in terms of efficiency. They remain consistent and within the *Highly Desirable* interval of preference across all configurations.

Comparison between the Physical Programming Methods

The main difference between the three methods lies in the complexity of the class function's formulation in order to obtain a more or less smooth curve, so that the class functions translate the preferences of the Decision Maker as accurately as possible. The original NLPP formulation allows the most accurate depiction of the Decision Maker's preferences, with a smooth curvature, but at the cost of very complex formulations. The GPP method simplifies the formulation while keeping the Physical Programming properties. Finally the LPP method allows a linear formulation of the class functions, which not only reduces the computational complexity, but also allows its use with traditional linear solvers that are more efficient than the proposed exact algorithm.

In light of the results that we have obtained from our different simulations, we were able to determine that the three methods provided similar results, but the Linear version slightly outperformed the other methods with a difference of efficiency of $2 - 3\%$. Therefore, we can conclude that the LPP method is more suitable for this problem, as it is simpler in terms of computational complexity, and provides results that are closer to the preferences of the designer.

Comparison between the Exact Solutions and the Heuristic

The evaluation of the heuristic shows its efficiency with results that, on average, remain within the desirable intervals, while exploring only parts of the solution space, and with limited visibility on the domains. However, its efficiency decreases as SFC length increases, especially in terms of bandwidth per user. In terms of runtime, the exact solutions show relatively high runtimes that quickly increase as the SFC length and topology size increase, thus making their use unsuitable for real time SFC placement. On the other hand, the heuristic solution proves to be more scalable with significantly lower computation times. This can be explained first by the fact that the heuristic explores a smaller set of solutions, and second by the SFC partitioning process that takes place, thus further reducing the number of possible combinations for each sub-SFC. Furthermore, the runtime for the heuristic can be controlled by increasing or decreasing the number of generations for which the algorithm is run, and a fair trade-off between efficiency and runtime can be obtained depending on the Decision Maker's priorities.

4.7 Conclusion

In this chapter, we proposed a centralized framework in order to support SFC placement over multiple domains while allowing the domain operators to disclose minimal information on their infrastructure. We modeled the SFC placement problem as a multi-objective optimization problem where the end-to-end latency, individual bandwidth, and relative deployment cost are optimized using Physical Programming in order to express preferences through meaningful parameters. Three Physical Programming methods were used: Linear, Non Linear, and Global. We implemented our model using an exact algorithm, and also formulated a scalable and efficient heuristic solution. The evaluation of the framework as well as the proposed algorithms proved our solution's effectiveness, with limited visibility on the network, and the scalability of the heuristic that provided results within the desirable ranges in relatively low runtimes. Furthermore, we were able to demonstrate the benefits of using Physical Programming as opposed to the weighted-sum method by comparing our results to those obtained from an ILP, indeed, the results showed that the Physical Programming method provided solutions that optimized all of the objectives, as opposed to the ILP. The comparison between the three Physical Programming methods also provided insights on the results of each one.

In the next chapter, we detail a scalable multi-domain SFC deployment and orchestration framework that ensures the end-to-end SFC forwarding, and manages the differences in encapsulation between the domains.

Chapter 5

SFC Deployment and Orchestration

5.1 Introduction

In the previous chapters, we studied multi domain SFCs mapping with limited visibility, and provided a framework that allows orchestrators to determine the optimal placement of the SFCs while satisfying a set of deployment constraints. However placement is only a part of the SFC deployment process, and is not sufficient for ensuring the effective end-to-end deployment, as it requires additional configuration steps. Indeed, to ensure the end-to-end forwarding of SFC packets, the domain orchestrators need to identify the SFC that ingress packets belong to, as well as the next hop, to properly configure their forwarding and classification components. This presumes that a communication protocol between orchestrators ought to be in place in order to enable the sharing of the required information. Furthermore, to support heterogeneous encapsulation types in the local domains, an interfacing entity should be deployed to apply the required encapsulation at the ingress of the domain and to add the encapsulation that would allow the next domain to identify the SFC at the egress of the domain.

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

5.1.1 Multi-Domain SFC Placement

As shown in the previous chapters, a number of contributions [140,143,145,170–172] studied SFC placement with limited visibility, with two main architectural approaches. With the first approach, a distributed algorithm is executed on all of the 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 the information disclosed by the domains, and

reconstitutes an abstract global view of the network in order to perform placement. In the chapters 3 and 4, we proposed centralized solutions for SFC placement. However, these works only solve 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 configuration steps.

5.1.2 Multi-Domain Service Orchestration

The ETSI framework for NFV-MANO has been extended in its third release [135] 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 NSDs. 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 a position paper, Rosa *et al.* [173] proposed a multi-domain NFV orchestration architecture, presented a few use cases, and identified challenges and open research directions such as control plane placement, resource capability discovery, policy enforcement, and security. In [174], 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 [175]. Figueira *et al.* [141] proposed a hierarchical architecture for multi-domain SDN Control and orchestration. The framework comprises a main domain that contains the main orchestration and control components, and that interfaces with lower level domains (Region and leaf domains). [142] presented the 5G Exchange framework for end-to-end multi-domain orchestration. Single domain orchestrators of a single operator are connected to an upper-level logically centralized multi-domain orchestrator, which communicates with other operator's multi-domain orchestrators in a distributed manner. However, these works don't consider SFC and the related constraints on the packet processing order.

5.1.3 SFC Orchestration

Multiple contributions proposed SFC orchestration solutions that extend the ETSI MANO [52] reference architecture. The authors of [60] introduce ETSO, a modular ETSI NFV-MANO compliant framework that ensures an end-to-end SFC orchestration. Likewise, Medhat *et al.* [62] detail X-FORCE, an ETSI NFV-MANO compliant SFC orchestration framework that integrates an SFC orchestrator in the reference architecture, and Ding *et al.* [36] 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. A few works tackled the multi-domain SFC deployment issue: In [176], 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 [177] proposes two methods for inter-data-center SFC deployments:

- **With multiple SFC domains:** Where each domain's management is constrained to itself, meaning that each SFC domain is only aware of the sub-SFC that is deployed within its infrastructure, as well as the next hop, and is unaware of the higher-level SFC details and configuration, as well as the other SFC domains. The SFC is fragmented, and a sub-chain is deployed in each domain. In this scenario, classification needs to be performed at the entry of each sub-chain, which increases delays; this scenario also doesn't allow context data sharing.
- **With a single SFC domain:** In this scenario, classification is performed only once and context data can be shared, but control and orchestration complexity increases as all domains have to be managed at once by a single logical entity. Furthermore, 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, as stated previously.

5.1.4 SFC Packet Forwarding

As for SFC packet forwarding, multiple traffic steering methods have been proposed, such as header-based methods, like Network Service Headers (NSH) [41, 178], tag-based methods [179] [180], and programmable switch-based methods [181]. A comprehensive survey on traffic steering methods for SFC can be found in [30]. However, these solutions enable SFC packet forwarding inside a single domain only. The RFC 8459 [27] 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. The proposal also introduces the IBN that acts as an SFC-aware Service Function 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 interoperability. 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. Another draft in [136] 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.

In this chapter, we tackle the aforementioned issues by presenting a novel ETSI-compliant 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. First, we propose an architectural framework 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. Afterwards, we implement a Proof of Concept of our architecture, and perform an extensive evaluation based on various KPIs using different encapsulation protocols.

5.2 Architecture

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 [52] [135], as well as the hierarchical SFC principle proposed by the IETF [27]. We assume 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, SLA 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 5.1 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 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.

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 and WAN Infrastructure Manager 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, additional information that isn't supported by that reference point (see Section 5.4).

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 5.3. 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 [27] 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 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:

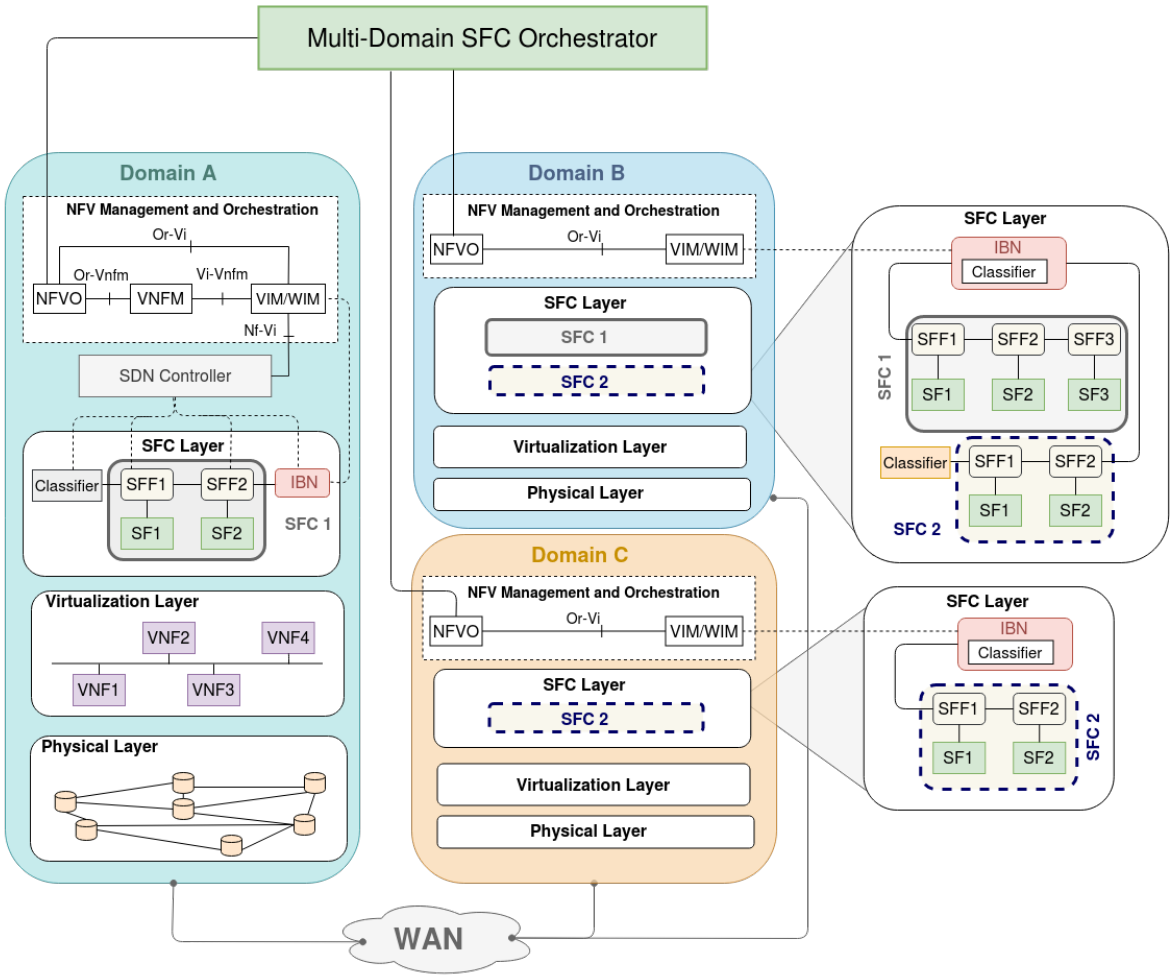


Figure 5.1: Architecture for Multi-Domain SFC Orchestration

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. 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 5.4. This mapping table is constructed using information obtained from the MDO, and the local NFVO through the 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.

5.3 Multi-Domain SFC Instantiation Process

We hereby describe the SFC Instantiation process over multiple domains. Figure 5.2 illustrates the workflow that takes place when instantiating a multi-domain SFC. At first, the customer, who wishes to deploy its SFC, sends the corresponding Network Service Descriptor (NSD) to the multi-domain orchestrator (see chapter 3); in turn, the latter will perform an initial placement of the

chain, which 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 5.3.1. 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 5.3.2.

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 NSH encapsulation [41], the path ID would correspond to the Service Path Identifier, and the position in the path would correspond to the Service Index. 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, 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.

5.3.1 Request Placement

As previously explained in Chapter 3, 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 explored in the previous chapters.

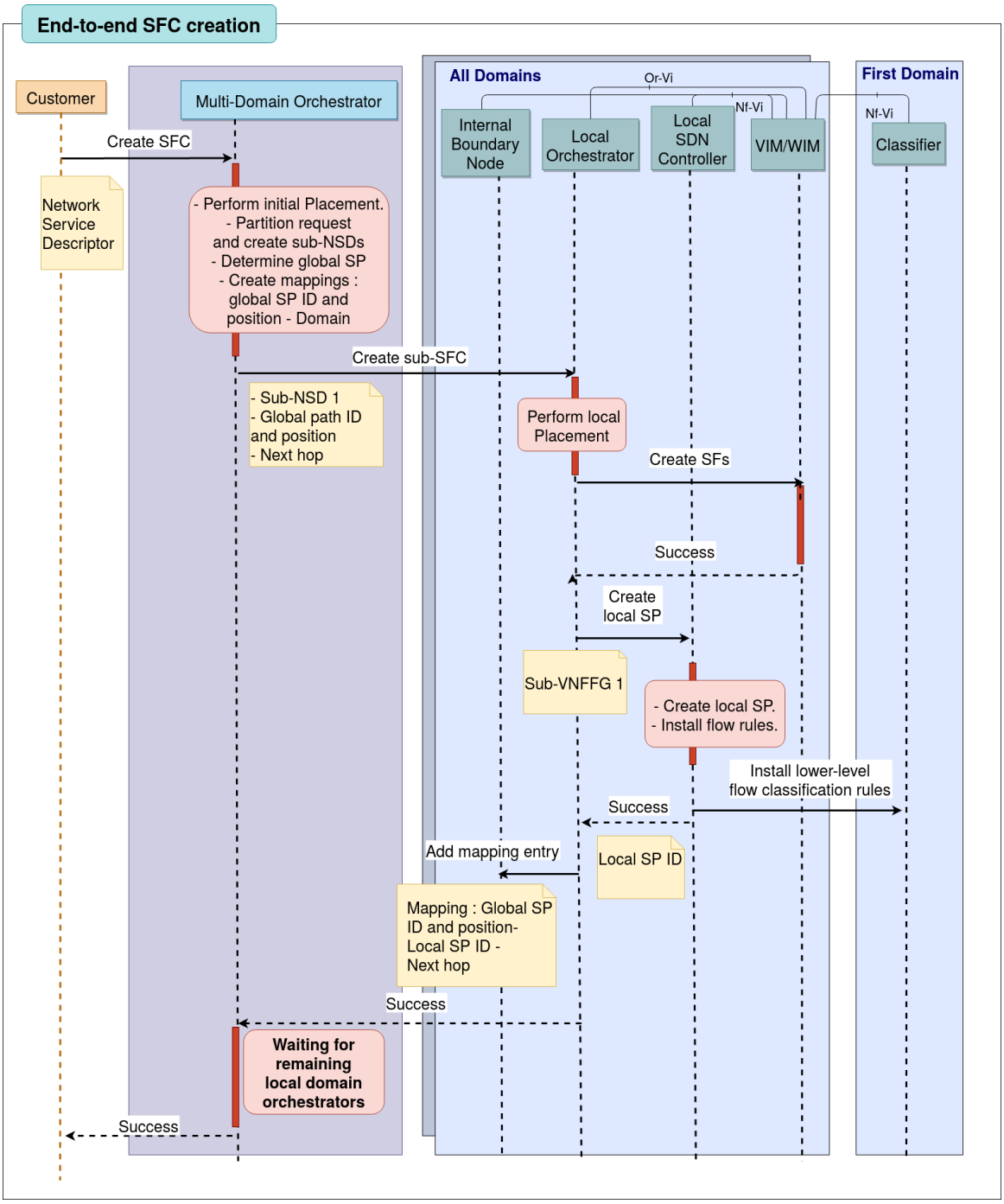


Figure 5.2: Multi-domain SFC instantiation workflow

5.3.2 Request Partitioning

NSD partitioning is performed as follows: The consecutive VNFs that have been placed on the same domain are grouped, in order to create sub-SFCs, with connection points 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 detailed in figures 5.3, 5.4, and 5.5.

In this example, we provide two scenarios 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 [182], 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 5.3, 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.

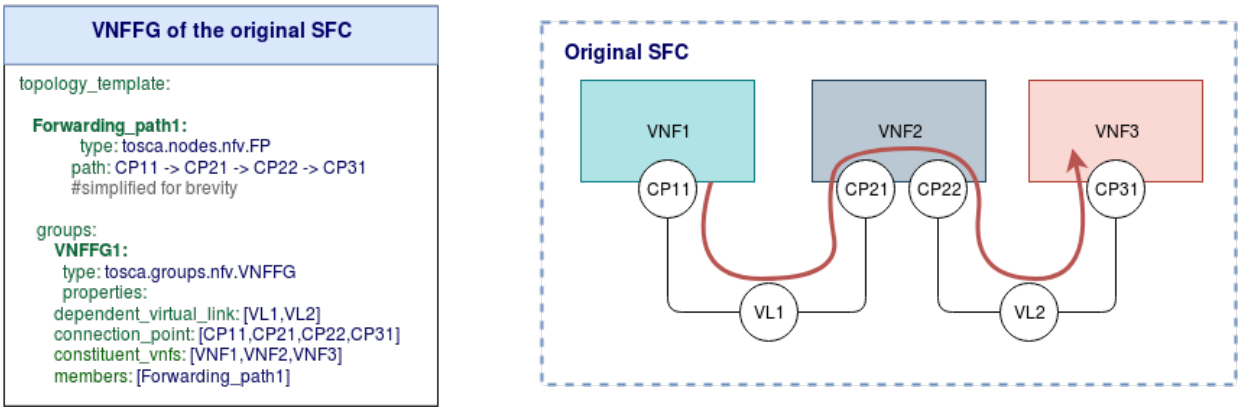


Figure 5.3: Original SFC and its VNFFG before partitioning

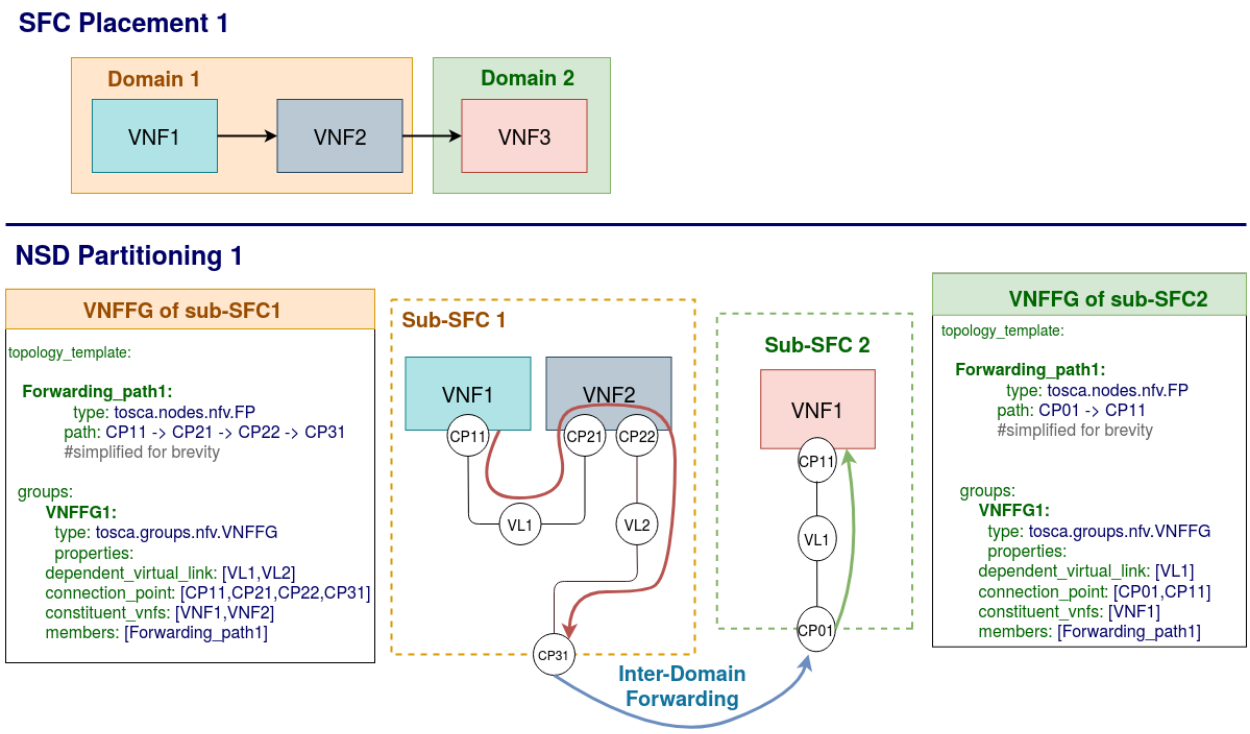


Figure 5.4: Sub-SFCs and their VNFFGs for the first partitioning option

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 Figures 5.4 and 5.5: 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. 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.

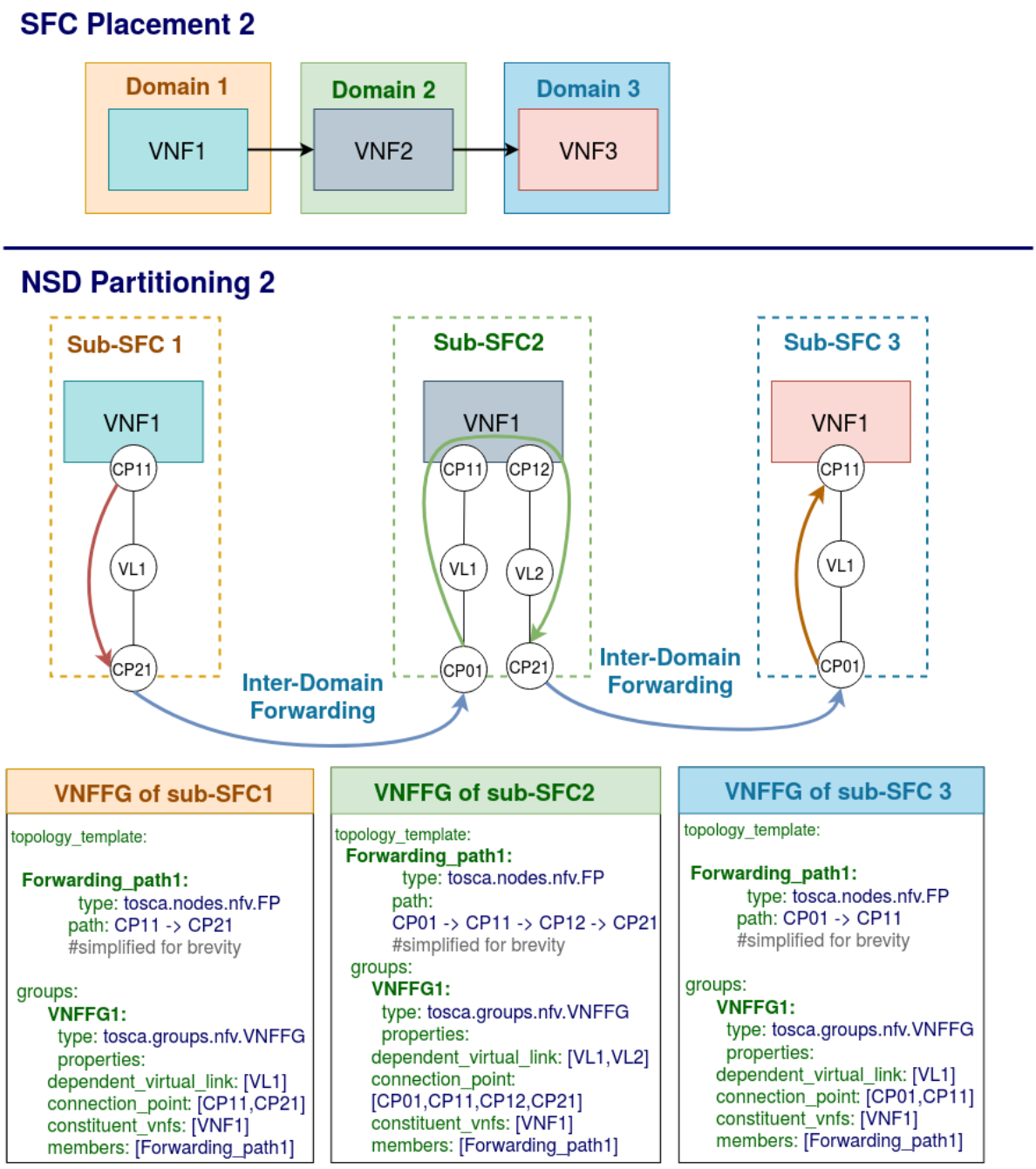


Figure 5.5: Sub-SFCs and their VNFFGs for the second partitioning option

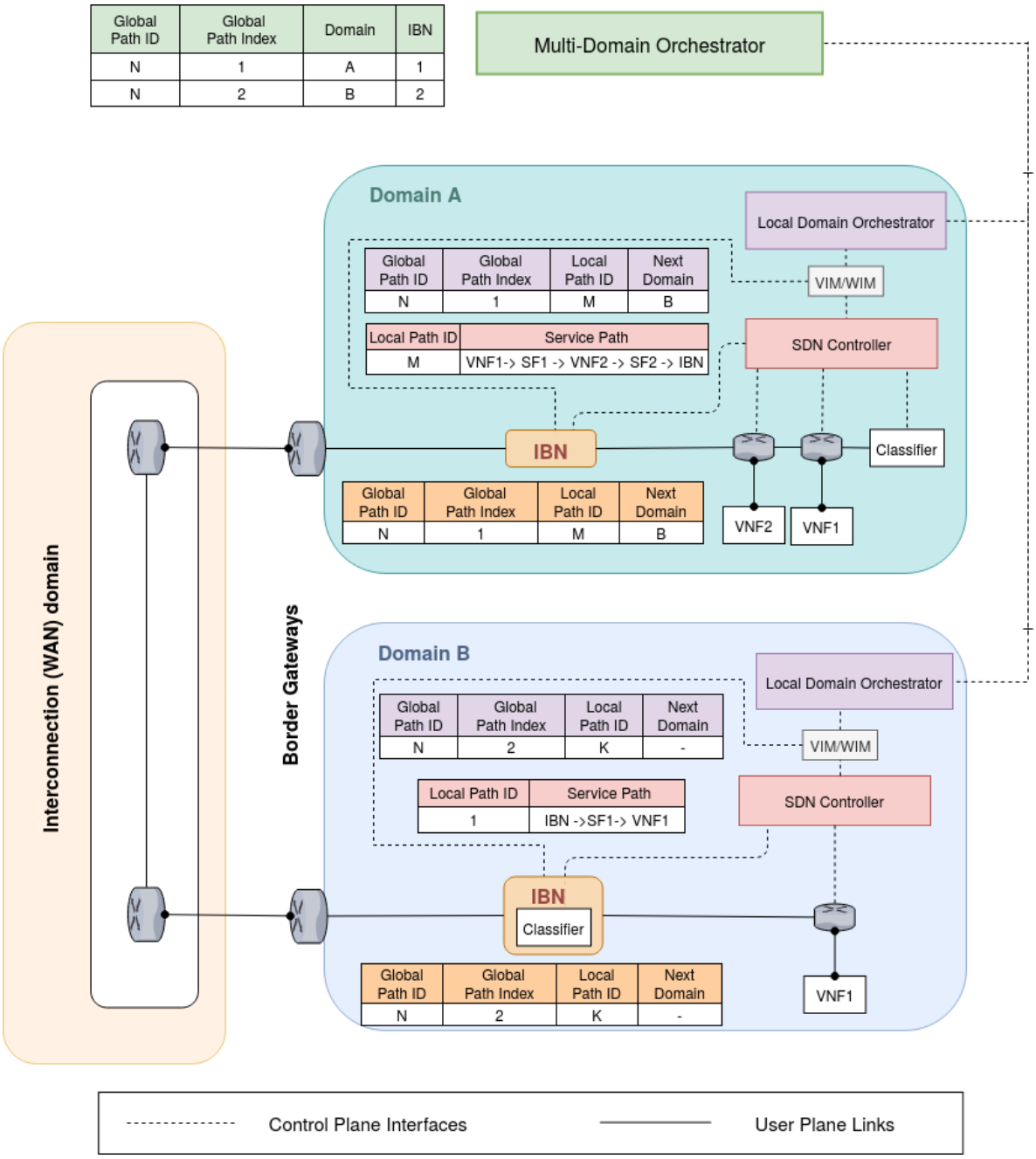


Figure 5.6: End-To-End SFC Packet Forwarding Mechanism

5.4 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). Figure 5.6 shows the color-coded mapping tables of the different architecture components in our framework 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 installed 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 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:

The WAN domain is SFC aware: In this case, the IBN re-encapsulates the packets using the higher level (WAN level) SFC encapsulation, and the border gateways as well as the WAN routers act as higher level SFFs and forward the packets to the IBN of the next domain. 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 (border gateways and WAN routers) would need to be connected to a control plane that manages the higher level chains, such as a 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 need 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 local 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 of the VNFs

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:

- By determining 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.
- By including 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.

5.5 Multi-Domain SFC Deletion Process

In the following, we detail the deletion process of the multi-domain SFC. Figure 5.7 depicts the associated workflow. The process is triggered by the customer's request to delete the SFC that is sent to the MDO. Using the SFC ID and the previously described mapping table, the MDO is able to retrieve the corresponding sub-SFC IDs, as well as the domains where the sub-SFCs have been deployed. The MDO would then send deletion requests to each local domain with the associated sub-SFC IDs.

Upon receiving that request, each local domain orchestrator instructs its local VIM and SDN controller to stop and delete the VNFs of the SFC, and the associated classification and forwarding rules, respectively. Finally, through the WIM, each local orchestrator deletes the SFC's associated mapping entry from the ingress and/or egress IBN's mapping table, then sends a confirmation to the MDO. After receiving the successful deletion confirmation from all of the local orchestrators, the MDO concludes the process.

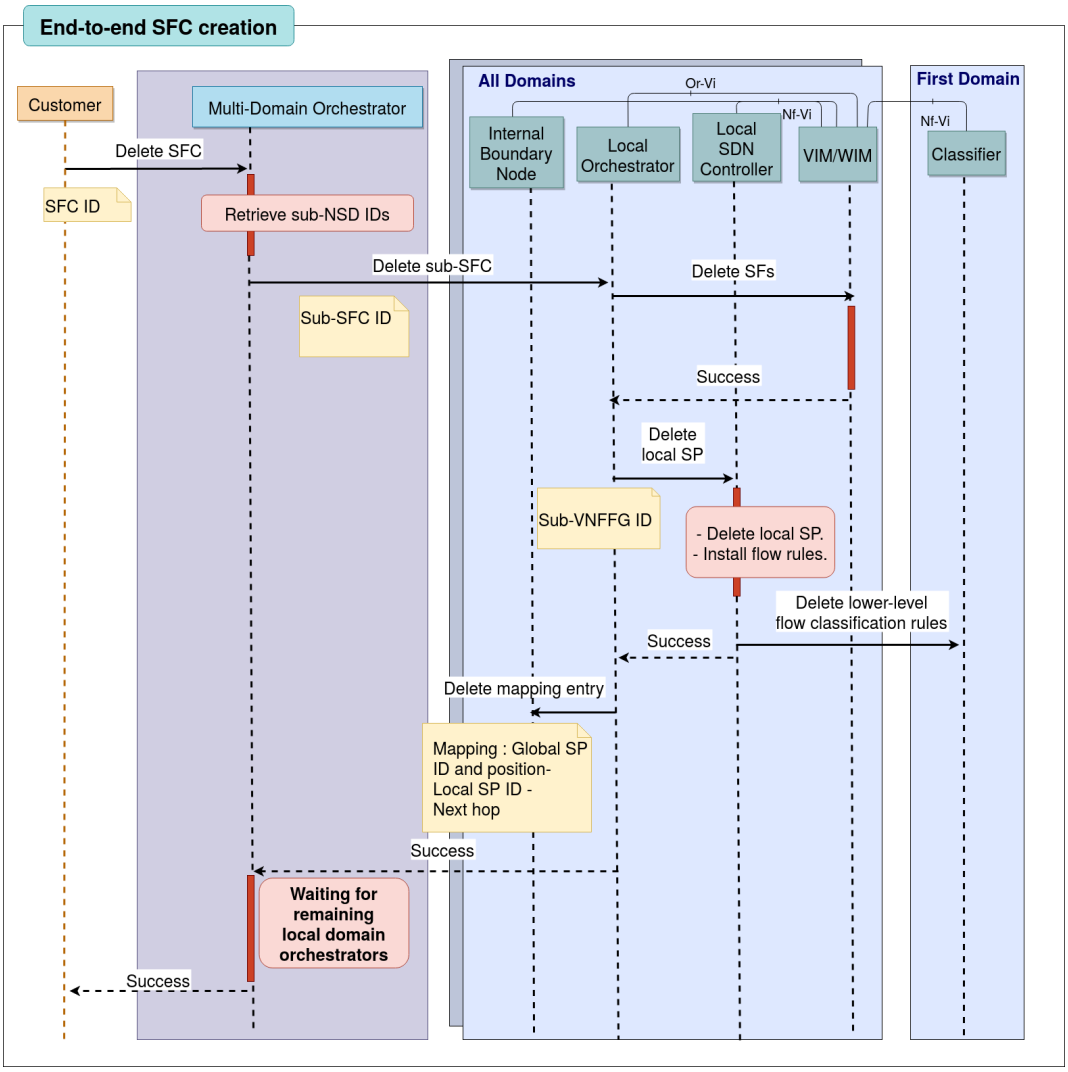


Figure 5.7: Multi-domain SFC deletion workflow

5.6 Proof of Concept Implementation

Next, to validate our framework, we implement 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.

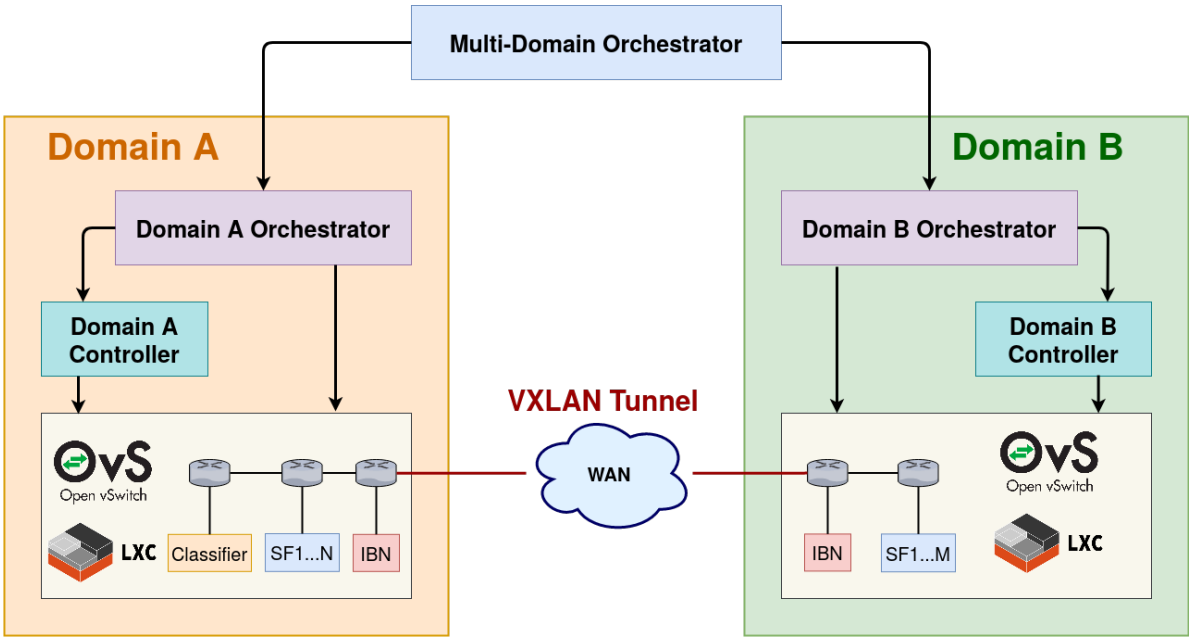


Figure 5.8: Proof of Concept Testbed Implementation

Testbed Setup

In the following, we depict our Proof of Concept of the proposed framework. As illustrated in Figure 5.8, 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 [183] containers that are used in order to host the different Service Functions connected through OVS (Open vSwitch) [184] switches that 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 in the classifiers, IBNs and VNFs was developed using the Python-based library *scapy* [185], and the traffic is generated using the *hping3* packet assembler [186]. 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 constructs the classification and forwarding rules for each SFC, and enforces them at the switch level, at the classifier's level, and in the IBNs at the domain's borders. The hardware and software setup is outlined in Table 5.1.

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 5.1: Testbed Hardware and Software Configuration

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

- The Network Service Header [41]. 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.

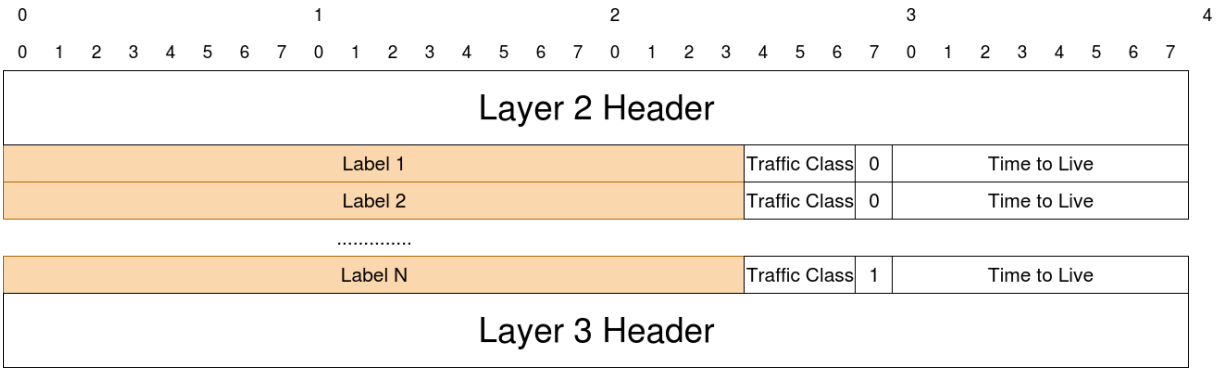


Figure 5.9: MPLS-Based Segment Routing Encapsulation

- Segment Routing [39] using Multi Protocol Label Switching (MPLS) headers as depicted in Figure 5.9, 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. 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 and deletion time of the end-to-end SFCs. The deployment time is measured 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 previous chapters, furthermore, a larger multi-domain topology would be needed to obtain meaningful results. Additionally, we evaluate the SFC deletion time, from the reception of the deletion request from the user, to the effective deletion of all of the SFC elements and configurations. 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.

5.7 Results

5.7.1 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, 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 5.10 features the average deployment times as well as the 95% Confidence Interval (C.I). Further, 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

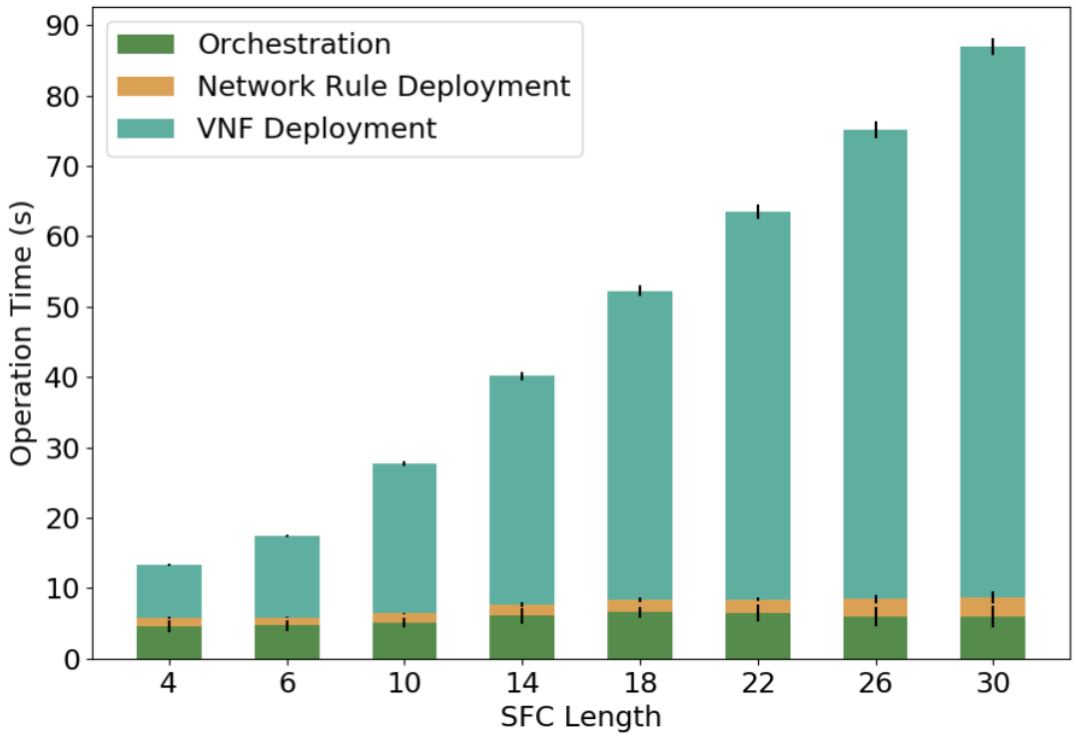


Figure 5.10: End-to-End SFC Deployment Time

Service Function, Classifier and IBN configuration by the local domains; while the orange bars represent the 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 5.2.

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. 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*

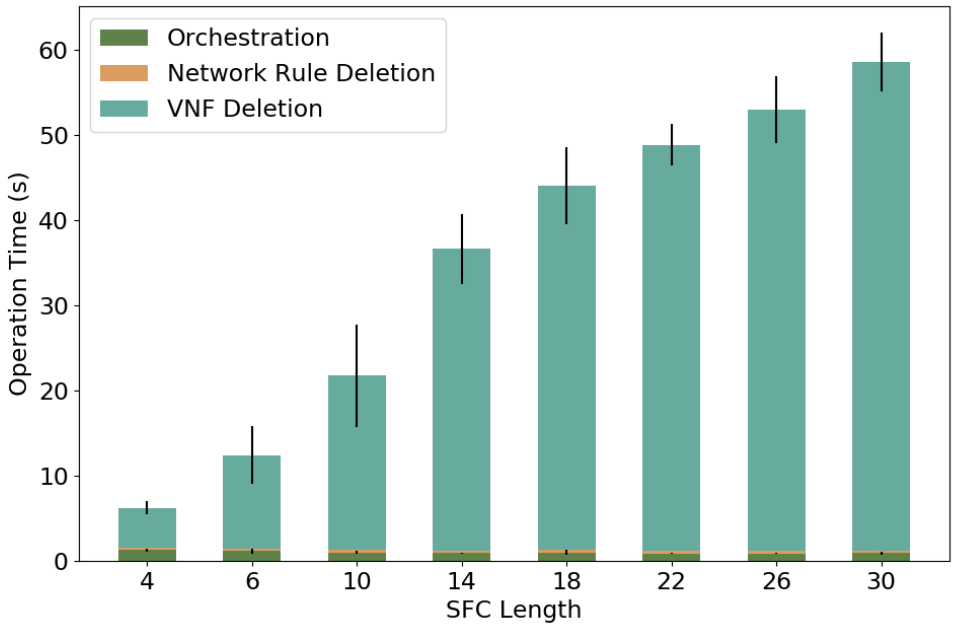


Figure 5.11: End-to-End SFC Deletion Time

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 parallelly 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.

5.7.2 Deletion Time

Following the successful deployment of each SFC described above, we trigger the deletion process. Similar to the previous experiment, we perform the evaluation for each SFC length 20 times. We provide in Figure 5.11 the mean and *95%* C.I values for each SFC length. The green bar represents the mean orchestration time, and the orange and blue bars represents the flow rule deletion time

and the VNF deletion time respectively. The summed values of the three bars represent the total multi-domain SFC deletion time.

Similar to the deployment time, it can be seen that the deletion times increase proportionally to the SFC length, with values ranging from *6.22s* to *58* seconds for SFC length of 30, which is lower than the deployment times. This difference is mainly due to the orchestration time and network rule deletion times that are significantly lower, and that remain stable across SFC lengths with values of *800ms* to *1.27s*, and *273ms* to *291ms*, respectively. In contrast, the VNF deletion times are similar to the VNF deployment times, with values ranging between *4.69s* for the SFC length of 4 VNFs, and *57.4s* for the SFC length of 30, thus making up for *75-98%* of the total deletion time.

5.7.3 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 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. Therefore, we generate the SFPs, and the corresponding traffic steering rules for the virtual switches as follows:

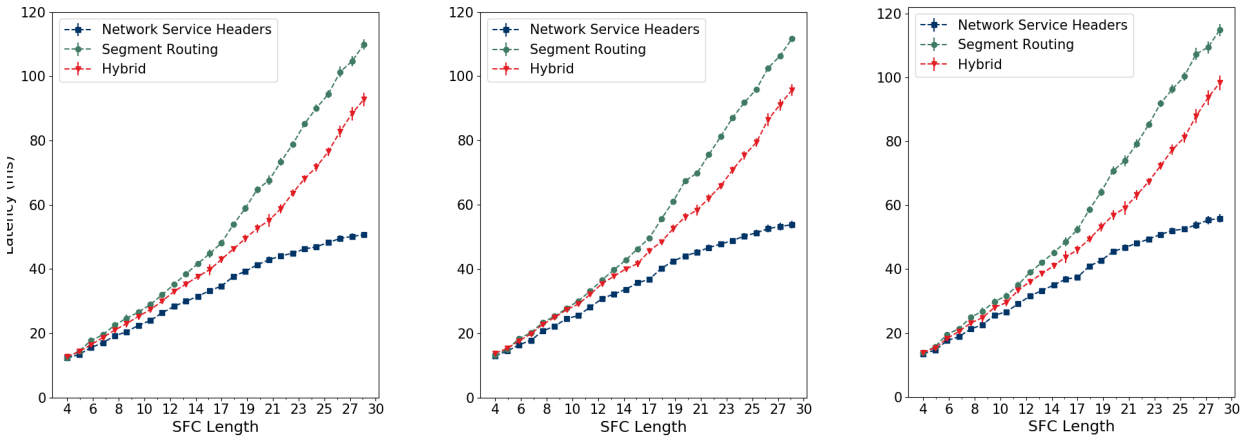
- **Segment Routing:** This encapsulation doesn't require SFP knowledge from the SFFs. As the list of hops is encoded by the classifier into the packet header in the form of MPLS labels. For each SFF, a rule is added to forward the packets to the connected SF if the top MPLS label comprises its ID, and otherwise, a second rule is added to forward the packets to the next SFF. Since the IDs of the SFs are the same, one set of rules can be used by all of the SFCs. For this simulation with 30 SFFs, 63 rules in total were needed, with an average of 2 rules per SFF.

- **Network Service Header:** With NSH, for each SFC, a different SPI and SI combination is assigned to the SFs, which means that each SFC should have a different set of forwarding rules to be enforced on the SFFs. Table 5.2 shows the total number of rules, the mean number of rules per SFF, and the rule deployment time for each number of SFC.

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. Note that different traffic flows are generated and that each SFC is classified using its IP and transport header fields.

Number of SFCs	Total Number of Rules	Number of rules per SFF
50	1533	51
150	4533	151
450	13533	451

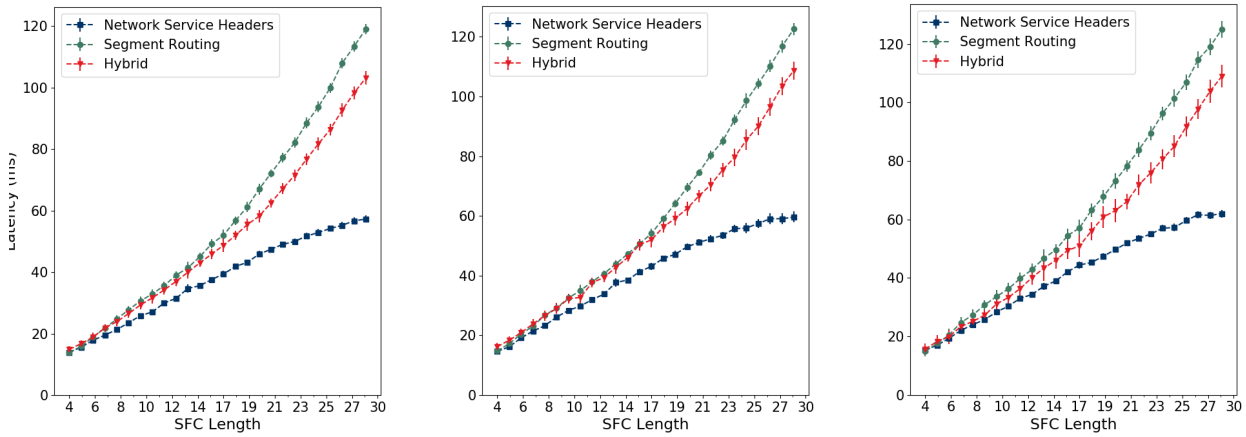
Table 5.2: Number of forwarding rules per number of SFCs for NSH



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

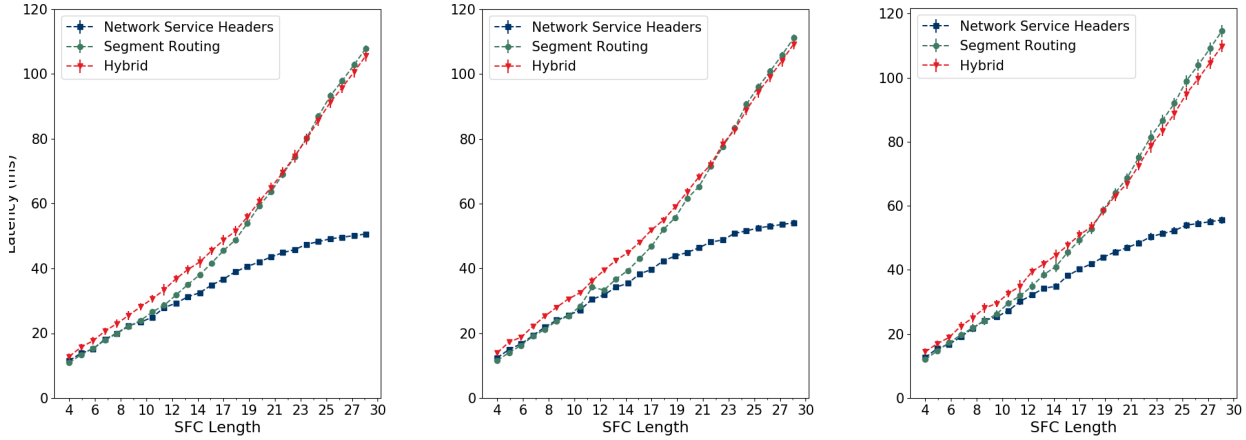
Figure 5.12: End-to-End Latency with 50 SFCs

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 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. Each SFC can be identified by the classifier using the packet headers and payload. Figures 5.12, 5.13, and 5.14 feature the measured latency in milliseconds as well as the 95% Confidence Interval for each SFC number and packet rate. The blue line represents the latency of the full-NSH scenario, the green line represents the latency of the full Segment Routing implementation, and the red line illustrates the latency for the hybrid scenario.



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

Figure 5.13: End-to-End Latency with 150 SFCs



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

Figure 5.14: End-to-End Latency with 450 SFCs

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 SFC number 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 450 SFCs, 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.

5.7.4 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, which corresponds to the process of adding encapsulation to the packets by the classifier, ingress and egress IBNs, and changing the packet encapsulation in order to forward it to the next hop by the SFs. This allows us to determine the latency added by the packet processing operations necessary in order to implement both encapsulation types. Figure 5.15 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, and 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 sub-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 5.1. 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, and 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.

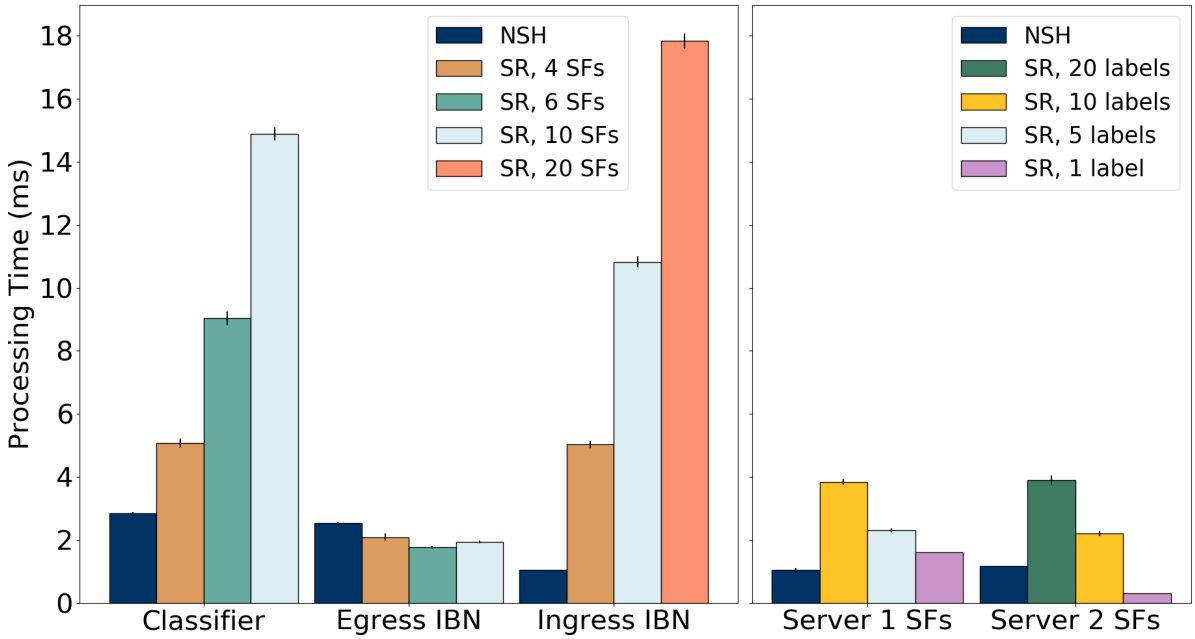
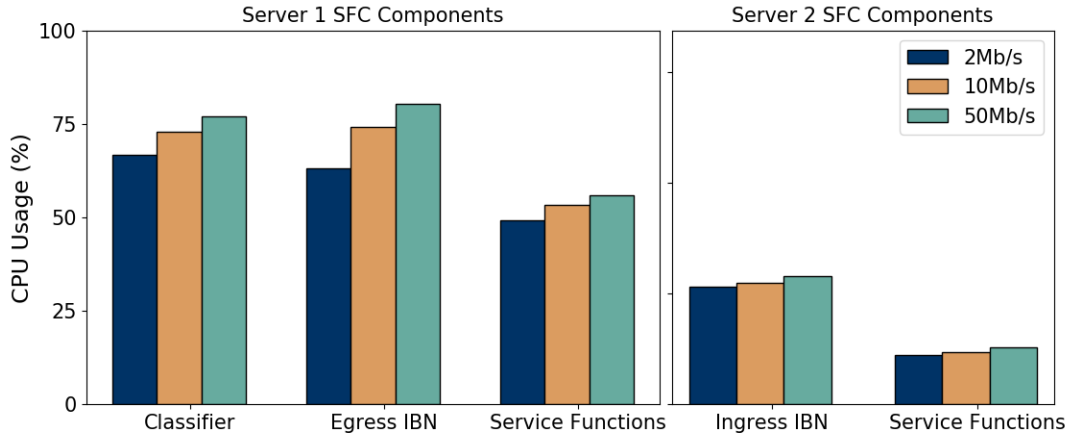


Figure 5.15: Processing Time per SFC Component

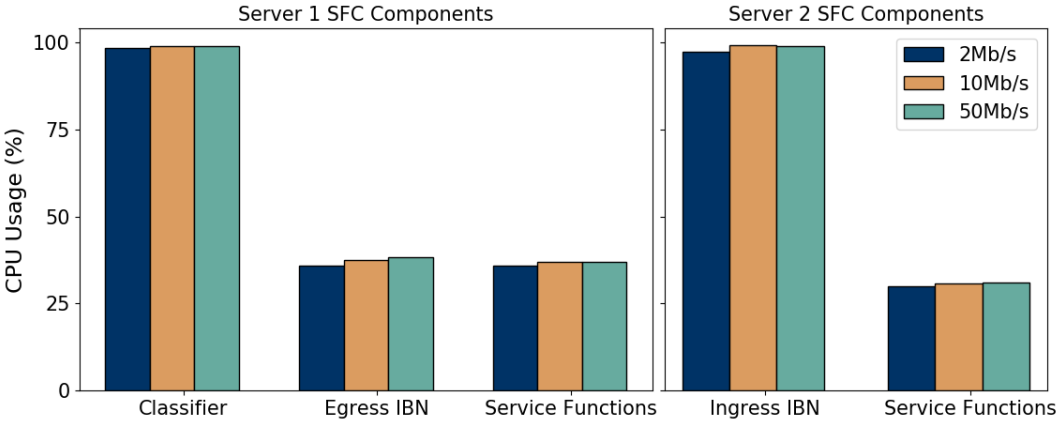
5.7.5 Processing Load

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. We considered two scenarios, one where the CPU load is probed when the SFC components are processing packets for all SFCs in the same run, similar to the previous metrics. For the second scenario, we generate traffic for only one SFC, which allows us to observe the effect of the SF position in the SFP on the CPU load. Indeed, in the previous scenario, since VNF sharing is enforced, a VNF might have different positions in the SFP of each Service Chain.

Figures 5.16a and 5.16b illustrate the CPU loads for the different components, for all of the deployed SFCs for NSH and Segment Routing respectively, while Figures 5.17a and 5.17b 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



(a) Encapsulation: Network Service Headers



(b) Encapsulation: Segment Routing

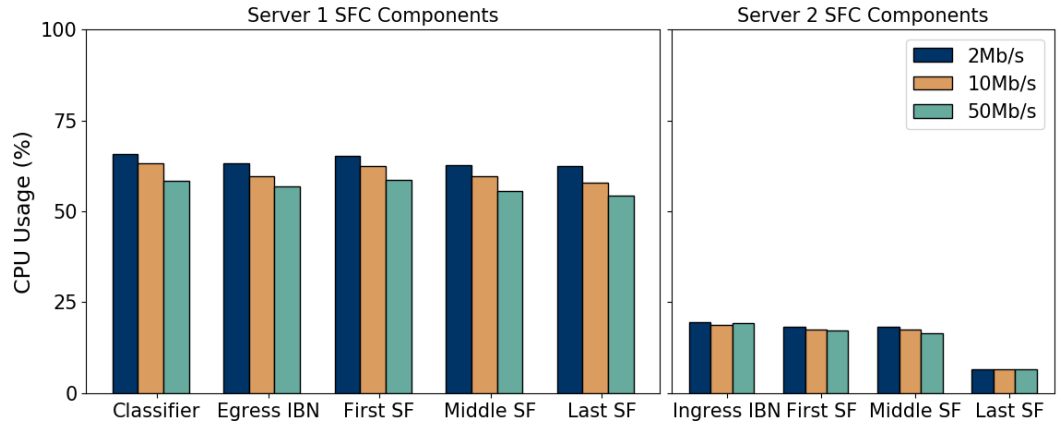
Figure 5.16: CPU Usage of the SFC components for all rates and all SFCs

the second part comprises the remaining 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.

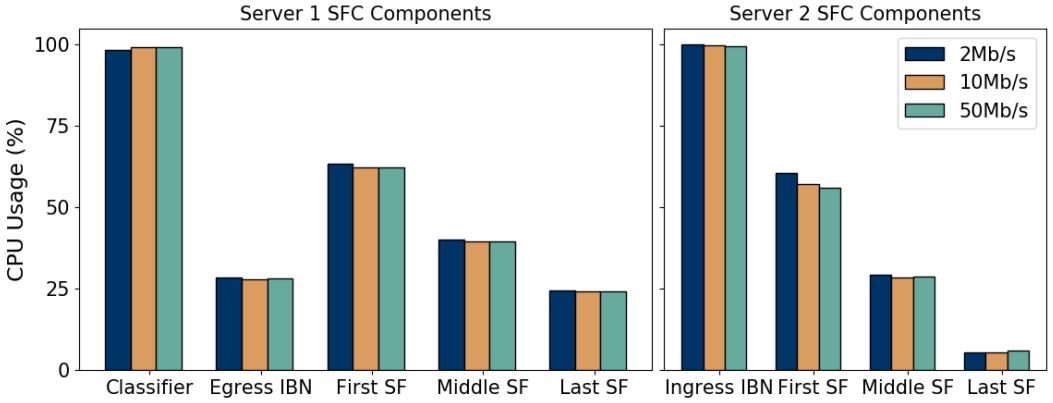
In all figures, and for all of the components and encapsulations, we can notice a slight increase in the CPU load when the traffic rate increases. It can also be observed that for the NSH encapsulation, the CPU load remains stable across components, while it fluctuates for Segment Routing depending on the component, its function, and position along the SFC. Looking at the CPU values for all SFCs in Figure 5.16, 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 5.17. 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 *41%* 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. Furthermore, the traffic rate didn't have a significant effect on the CPU usage for both encapsulations.



(a) Encapsulation: Network Service Headers



(b) Encapsulation: Segment Routing

Figure 5.17: CPU Usage of the SFC components for all rates, for SFC of 30 VNFs

5.7.6 Results Discussion

Scalability

By synthesizing the obtained results, it can be concluded that the workflow of our proposed framework enables an end-to-end deployment and deletion across domains with minimal additional time compared to the time consumed by VNF deployment and deletion at the VIM level. It is dependent on the hardware configuration of the infrastructure and the virtualization technology deployed by the operator. Moreover, the orchestration and network rule deployment/deletion times remain stable as SFC length increases, which proves the scalability of our framework regarding SFC length. As for the scalability related to the number of domains, since the deployment or deletion 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 time. The overall time would only be affected by the deployment or deletion time on the least efficient domain, since the MDO waits for the confirmation messages of all of the local orchestrators before concluding the process.

Encapsulations

Comparing the different results for the NSH and Segment Routing encapsulations, and looking at the end-to-end latency, 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, we can observe 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, because the Segment Routing encapsulation size decreases along the chain as the MPLS tags are popped at each hop, this explains why the greatest 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-SFCs that need to be deployed by each domain: a shorter sub-SFC may benefit from the Segment Routing encapsulation as the number of forwarding rules, and total encapsulation size would remain minimal, thus ensuring a lower traffic overhead, and lowering the amount of stored forwarding rules in the SFFs. However, the NSH encapsulation should be considered for longer SFCs despite the number of rules, since the Segment Routing encapsulation 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 5.7.5 were consistent with the previous metrics, where the CPU usage for NSH was similar for the SFC components, with relatively low values, while for Segment Routing, the CPU usage decreased as packets were 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.

5.8 Conclusion

In this chapter, we introduced a novel framework that enables multi-domain SFC deployment. The architecture is ETSI-MANO compliant, and leverages the hierarchical SFC principle using the IBN as an interfacing entity for the local domains. We also detailed the SFC instantiation process, and 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. We assessed its effectiveness with a Proof of Concept implementation that has been evaluated by measuring different Key Performance Indicators for SFCs. The obtained results demonstrated our framework's scalability and efficiency, which allowed us to draw conclusions related to the most suitable encapsulation type depending on SFC length.

Chapter 6

Conclusions and Perspectives

6.1 Conclusions

The new 5G networks are set to enable multiple heterogeneous use cases such as Augmented Reality, autonomous vehicles, or IoT, with different requirements such as a very low latency, or support for a high density of connected devices. The average throughput per user is also significantly increased, thus allowing the streaming of high quality videos. In order to satisfy all of these requirements, multiple technologies are developed such as Software Defined Networks, Network Function Virtualization, Network Slicing, or Service Function Chaining. This latter is a concept by which, to deliver an end-to-end service, the packets are steered in an ordered manner through a set of functions that perform specific processing operations.

In some cases, the components of a Service Function Chain are required to be placed on more than one domain, for multiple reasons such as the lack of resources, the mobility of users, or for functional or security considerations. The multi-domain context introduces a set of new challenges, in particular, the lack of visibility on the domain's infrastructure makes the placement procedure more difficult, and the end-to-end packet forwarding needs to take into account the possible heterogeneity in the packet encapsulations that are employed by each domain. Besides, the independence between the domains in terms of orchestration and management introduces complexity in the life-cycle management operations of the multi-domain SFC.

This thesis investigates different solutions to achieve multi-domain SFC. In particular, we tackle the placement issue with a limited visibility, with a joint optimization of multiple QoS metrics as well as the deployment cost for different SLA classes. We model the problem's constraints

and objectives, and propose exact solutions and heuristics. Further, we elaborate a scalable multi-domain SFC orchestration framework that leverages on ETSI and IETF standards. The framework enables SFC deployment, end-to-end packet forwarding, and deletion, and the framework is also agnostic to the encapsulation type that is employed by each local domain.

The contributions of this thesis can be summarized as follows:

- In Chapter 3 we present a hierarchical placement scheme with limited visibility over multiple domains that supports complex non-linear SFCs, with a backtracking mechanism to handle local placement failure. Additionally, we model the multi-domain SFC placement as a multi-objective ILP, then propose a scalable and efficient memetic algorithm that performs SFC mapping according to the client's SLAs.
- The work in Chapter 4 elaborates a more accurate formulation of the user's preferences using three Physical Programming approaches (Linear, non-linear, global). Three optimization objectives are consideredo: the end-to-end latency, the bandwidth per user, and the overall cost. We propose an algorithm to find the exact solution based on a Branch and Bound approach as well as a scalable heuristic algorithm.
- In Chapter 5, we propose an architectural framework 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. Afterwards, we implement a PoC of our architecture, and perform an extensive evaluation based on various Key Performance Indicators (KPI) using different encapsulation protocols.

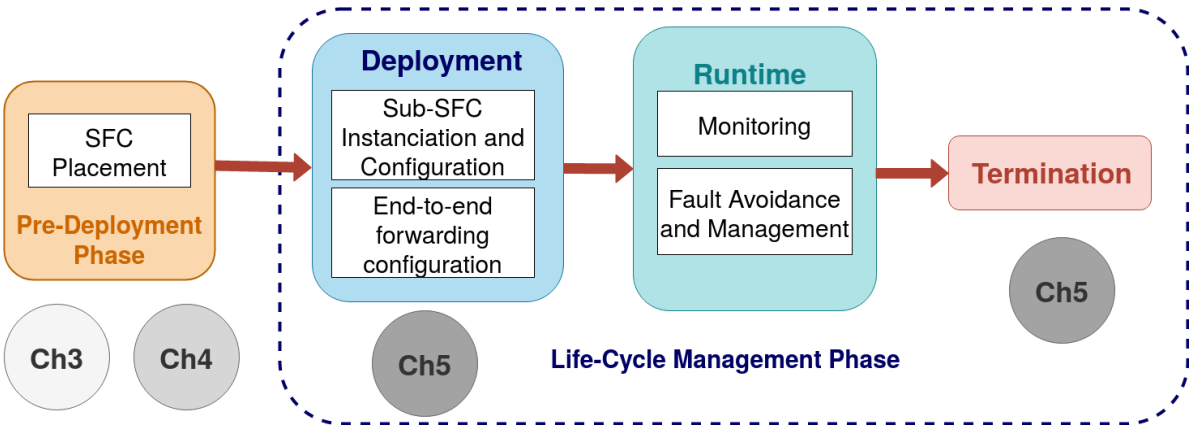


Figure 6.1: Contributions Mapped to the Multi-Domain SFC Life-cycle

Mapped to the multi-domain SFC life-cycle as shown in Figure 6.1, these contributions form a system that manages multi-domain SFCs during the pre-deployment and deployment stages as shown in 6.1. Indeed, the placement algorithm, and the SFC partitioning scheme can be incorporated to the Multi-Domain Orchestrator described in Chapter 5, in order to determine the optimal placement scheme that allows the MDO to perform the NSD partitioning. Therefore, this thesis provides a complete deployment mechanism for multi-domain SFC deployment, from the reception of the original request in form of an NSD by the MDO, until the effective deployment and configuration of the sub-SFCs on the local domains, as determined by the placement scheme. Furthermore, the proposed architecture can be extended in order to support life-cycle SFC management operations post-deployment.

6.2 Future Perspectives

In this thesis, we propose solutions for the multi-domain SFC placement and deployment. However, some challenges related to cross-domain SFC implementation remain untackled, and new research directions are worth exploring. In the following, we outline some future contribution perspectives.

End-to-End Life-cycle Management

In this work, we propose solutions for multi-domain SFC placement and 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 monitor and perform life-cycle management operations on its service chain at runtime.

Monitoring

Post-deployment, the SFC tenants should dispose of an interface that allows them to monitor their SFC, as well as the KPIs of their service, and the resource consumption of its components. Further, should a problem occur, the tenant should be able to determine the source of the problem.

Fault Management and Elasticity

Several 5G use cases are characterized by the mobility of users, such as autonomous vehicles, and e-Health. In both use cases, cameras and/or sensors capture information on the environment/patient and send that data to the cloud for processing, and pre-processing functions are deployed on edge clouds. However, since the users are in constant mobility, they might move away from the edge

cloud, far enough to increase the latency past the accepted limit. In that case, the operator might have to migrate the edge VNFs to edge clouds that are closer to those users, thus re-computing the SFC placement post-deployment.

Furthermore, in the context of fault management, cases of link or node failure would prompt the operator to re-compute the optimal link and node mapping of the affected SFC, and carry out flow re-routing, VNF migration or VNF re-instanciation. These operations can be performed locally by the domain orchestrator, or by the MDO that also gives instructions to the different local orchestrators. This raises the question of the scope of actions of the local orchestrators. Therefore, it should be clearly determined when the local orchestrators can make decisions and act independently, and when the MDO or the tenant (through the MDO) should be notified.

Additionally, 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. Indeed, 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 (see chapter 5).

Machine Learning for SFC Management

Recently, Machine Learning has gained popularity as a result of the availability of large sets of data, as well as the improvement in computational capabilities. Indeed, with the help of sufficient data, models can be trained in order to efficiently solve multiple problems. Machine Learning can be applied to the operations and management of networks in several ways: traffic prediction, classification and routing, and also congestion control, resource management, and fault management [187].

Multiple works have applied Machine Learning for SFC. In [188], Liu *et al.* propose a Quantum Machine Learning scheme for resource allocation in mobile edge SFCs. In [189, 190] Deep Learning is applied to traffic control in SDN/NFV enabled environments. Machine Learning is also used for SFC life-cycle management [191], and dynamic SFC routing [192].

Furthermore, Reinforcement Learning, which is a branch of ML, has gained a lot of attention [193]. With Reinforcement Learning, the agent leverages on the rewards from past experience and the environment's feedback in order to define a decision policy that would be continuously

improved at runtime. In [194–197], Reinforcement Learning is leveraged in order to perform SFC placement, it is also used in [198–200] for dynamic adaptive SFC placement, as well as traffic prediction and proactive resource allocation in [201].

Future works should investigate the benefits of applying the ML techniques for multi-domain SFC both in the placement and runtime operations, whether for the prediction of traffic fluctuations to proactively perform elasticity operations, or in the definition of policies for the multi-domain placement and life-cycle management while taking into account the limited visibility and control on the local infrastructure. These techniques would be

End-to-end 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. In turn the multi-domain orchestrator 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.

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 [27] 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.

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 examine traffic in order to avoid attacks and intrusions, then, each SFC component and link should be secured, indeed, 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.

Bibliography

[1] 3GPP, "Technical Specification Group Radio Access Network; Study on Scenarios and Requirements for Next Generation Access Technologies," 3rd Generation Partnership Project (3GPP), Technical Report 23.203, 07 2020, version 16.0.0.

[2] A. Al-Dulaimi, X. Wang, and C. L. I, *Standardization: The Road to 5G*, 2018, pp. 691–708.

[3] R. Vannithamby and S. Talwar, *5G Requirements*, 2017, pp. 9–22.

[4] 5GPPP, "5G Novel Radio Multiservice adaptive network Architecture (5G NORMA)," 5GPPP, Tech. Rep., 10 2015, Deliverable D2.1, Use cases, scenarios and requirements.

[5] 3GPP, "Technical Specification Group Services and System Aspects; Feasibility Study on New Services and Markets Technology Enablers," 3rd Generation Partnership Project (3GPP), Technical Report 22.891, 09 2016, version 16.0.0.

[6] N. Alliance, "NGMN 5G White Paper," NGMN Alliance, White Paper (WP), 03 2015.

[7] M. Iwamura, "Ngmn view on 5g architecture," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, 2015, pp. 1–5.

[8] ITU-R, "IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond," ITU-R, Tech. Rep., 09 2015, Recommendation ITU-R M.2083-0.

[9] C. Mannweiler, M. Breitbach, H. Droste, I. L. Pavón, I. Ucar, P. Schneider, M. Doll, and J. R. Sanchez, "5g norma: System architecture for programmable multi-tenant 5g mobile networks," in *2017 European Conference on Networks and Communications (EuCNC)*, 2017, pp. 1–6.

[10] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[11] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tut.*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[12] ETSI GS NFV 002, "Network functions virtualization (nfv); architectural framework v1.1.1," ETSI, Tech. Rep., October 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

[13] S. Newman, *Building Microservices*, 1st ed. O'Reilly Media, Inc., 2015.

[14] L. Chettri and R. Bera, "A comprehensive survey on internet of things (iot) toward 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

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

[16] Azure IoT Edge, https://azure.microsoft.com/en-us/services/iot-edge/.

[17] AWS IoT Greengrass, https://aws.amazon.com/greengrass.

[18] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. Abbas, "An in-depth analysis of iot security requirements, challenges and their countermeasures via software defined security," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[19] D. Martín-Sacristán, C. Herranz, and J. F. Monserrat, "Traffic safety in the metis-ii 5g connected cars use case: Technology enablers and baseline evaluation," in *2017 European Conference on Networks and Communications (EuCNC)*, June 2017, pp. 1–5.

[20] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2018.

[21] Google Cloud Functions, https://cloud.google.com/functions.

[22] AWS Serverless Computing, https://aws.amazon.com/serverless/.

[23] Azure Serverless Computing, https://azure.microsoft.com/en-us/overview/serverless-computing/.

[24] Apache OpenWhisk, https://openwhisk.apache.org/.

[25] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, *Serverless Computing: Current Trends and Open Problems*. Singapore: Springer Singapore, 2017, pp. 1–20.

[26] The State of Serverless Multi-cloud, https://www.serverless.com/blog/state-of-serverless-multi-cloud.

[27] 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

[28] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7665.txt

[29] I. Trajkovska, M.-A. Kourtis, C. Sakkas, D. Baudinot, J. Silva, P. Harsh, G. Xylouris, T. M. Bohnert, and H. Koumaras, "Sdn-based service function chaining mechanism and service prototype implementation in nfv scenario," *Computer Standards and Interfaces*, vol. 54, pp. 247 – 265, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092054891730017X

[30] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 487–507, Firstquarter 2019.

[31] NEC, VTN OpenFlow, https://wiki.opendaylight.org/images/d/da/NEC_VTN_Demo_0722.pdf.

[32] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula, "Steering: A software-defined networking for inline service chaining," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Oct 2013, pp. 1–10.

[33] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Dynamic chaining of virtual network functions in cloud-based edge networks," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–5.

[34] J. Blendin, J. Rückert, N. Leymann, G. Schyguda, and D. Hausheer, "Position paper: Software-defined network service chaining," in *2014 Third European Workshop on Software Defined Networks*, Sept 2014, pp. 109–114.

[35] S. Nirasawa, M. Hara, S. Yamaguchi, M. Oguchi, A. Nakao, and S. Yamamoto, "Application performance improvement with application aware dpn switches," in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Oct 2016, pp. 1–4.

[36] W. Ding, W. Qi, J. Wang, and B. Chen, "Openscaas: an open service chain as a service platform toward the integration of sdn and nfv," *IEEE Network*, vol. 29, no. 3, pp. 30–35, May 2015.

[37] H. Hantouti, N. Benamar, and T. Taleb, "Vlan-based traffic steering for hierarchical service function chaining," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[38] A. Abujoda, H. R. Kouchaksaraei, and P. Papadimitriou, "Sdn-based source routing for scalable service chaining in datacenters," in *Wired/Wireless Internet Communications*, L. Mamatas, I. Matta, P. Papadimitriou, and Y. Koucheryavy, Eds. Cham: Springer International Publishing, 2016, pp. 66–77.

[39] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," RFC 8402, Jul. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8402.txt

[40] ——, "Segment Routing Architecture," Internet Engineering Task Force, Internet-Draft draft-ietf-spring-segment-routing-15, Jan. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-spring-segment-routing-15

[41] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," RFC 8300, Jan. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8300.txt

[42] H. Zhang, L. Fourie, R. Parker, and M. Zarny, "Service Chain Header," Internet Engineering Task Force, Internet-Draft draft-zhang-sfc-sch-03, Dec. 2014, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-zhang-sfc-sch-03

[43] C. Jacquenet and M. Boucadair, " An IPv6 Extension Header for Service Function Chaining (SFC)," Internet Engineering Task Force, Internet-Draft draft-jacquenet-sfc-ipv6-eh-01, Jan. 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-jacquenet-sfc-ipv6-eh-01

[44] A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a linux-based nfv infrastructure," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–5.

[45] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[46] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 464–486, 2019.

[47] P. Quinn and J. Guichard, "Service function chaining: Creating a service plane via network service headers," *Computer*, vol. 47, no. 11, pp. 38–44, Nov 2014.

[48] M. Boucadair, "Service Function Chaining (SFC) Control Plane Components and Requirements," Internet Engineering Task Force, Internet-Draft draft-ietf-sfc-control-plane-08, Oct. 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sfc-control-plane-08

[49] G. Bernini, G. Landi, D. Lopez, and P. A. A. Gutierrez, "VNF Pool Orchestration For Automated Resiliency in Service Chains," Internet Engineering Task Force, Internet-Draft draft-bernini-nfvrg-vnf-orchestration-04, Apr. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-bernini-nfvrg-vnf-orchestration-04

[50] G. Davoli, W. Cerroni, C. Contoli, F. Foresta, and F. Callegati, "Implementation of service function chaining control plane through openflow," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–4.

[51] S. Kulkarni, M. Arumaithurai, K. K. Ramakrishnan, and X. Fu, "Neo-nsh: Towards scalable and efficient dynamic service function chaining of elastic network functions," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, March 2017, pp. 308–312.

[52] ETSI GS NFV-MAN 001, "Network functions virtualisation (nfv); management and orchestration," ETSI, Tech. Rep., December 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[53] ETSI GS NFV 001, "Network functions virtualisation (nfv); use cases," ETSI, Tech. Rep., October 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[54] ETSI GS NFV-EVE 005, "Network functions virtualisation (nfv); ecosystem; report on sdn usage in nfv architectural framework," ETSI, Tech. Rep., December 2015. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_NFV-EVE005v010101p.pdf

[55] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 533–546, Sept 2016.

[56] A. M. Medhat, G. A. Carella, M. Pauls, M. Monachesi, M. Corici, and T. Magedanz, "Resilient orchestration of service functions chains in a nfv environment," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 7–12.

[57] A. M. Medhat, G. A. Carella, M. Pauls, and T. Magedanz, "Orchestrating scalable service function chains in a nfv environment," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–5.

[58] L. Bondan, T. Wauters, B. Volckaert, F. D. Turck, and L. Z. Granville, "Anomaly detection framework for sfc integrity in nfv environments," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–5.

[59] W. Cerroni, C. Buratti, S. Cerboni, G. Davoli, C. Contoli, F. Foresta, F. Callegati, and R. Verdone, "Intent-based management and orchestration of heterogeneous openflow/iot sdn domains," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–9.

[60] 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, 2017.

[61] J. Wang, H. Qi, K. Li, and X. Zhou, "Prsfc-iot: A performance and resource aware orchestration system of service function chaining for internet of things," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2018.

[62] 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.

[63] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, Sept 2016.

[64] Y. Zhang, B. Anwer, V. Gopalakrishnan, B. Han, J. Reich, A. Shaikh, and Z.-L. Zhang, "Parabox: Exploiting parallelism for virtual network functions in service chaining," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '17. New York, NY, USA: ACM, 2017, pp. 143–149. [Online]. Available: http://doi.acm.org/10.1145/3050220.3050236

[65] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, "Nfp: Enabling network function parallelism in nfv," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 43–56. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098826

[66] W. Yan, K. Zhu, L. Zhang, and S. Su, "Efficient dynamic service function chain combination of network function virtualization," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, June 2017, pp. 163–168.

[67] Y. Wang, Z. Li, G. Xie, and K. Salamatian, "Enabling automatic composition and verification of service function chain," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, June 2017, pp. 1–5.

[68] A. F. Ocampo, J. Gil-Herrera, P. H. Isolani, M. C. Neves, J. F. Botero, S. Latré, L. Zambenedetti, M. P. Barcellos, and L. P. Gaspary, *Optimal Service Function Chain Composition in Network Functions Virtualization*. Cham: Springer International Publishing, 2017, pp. 62–76. [Online]. Available: https://doi.org/10.1007/978-3-319-60774-0_5

[69] J. Gil-Herrera and J. F. Botero, "A scalable metaheuristic for service function chain composition," in *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, Nov 2017, pp. 1–6.

[70] Y. Xie, Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," *CoRR*, vol. abs/1608.00095, 2016. [Online]. Available: http://arxiv.org/abs/1608.00095

[71] D. Ma, L. Zhuang, and J. Lan, "Efficient resource supplement for service function chaining in next-generation internet," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Oct 2016, pp. 2440–2445.

[72] H. Qing, Z. Weifei, and L. Julong, "Virtual network protection strategy to ensure the reliability of sfc in nfv," in *Proceedings of the 6th International Conference on Information Engineering*, ser. ICIE '17. New York, NY, USA: ACM, 2017, pp. 17:1–17:5. [Online]. Available: http://doi.acm.org/10.1145/3078564.3078583

[73] J. Fan, C. Guan, K. Ren, and C. Qiao, "Guaranteeing availability for network function virtualization with geographic redundancy deployment," 2015.

[74] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[75] T. Wen, H. Yu, G. Sun, and L. Liu, "Network function consolidation in service function chaining orchestration," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[76] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, June 2017, pp. 1–10.

[77] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, no. Part 3, pp. 492 – 505, 2015, cloud Networking and Communications II. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912861500359X

[78] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, Nov 2017.

[79] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[80] F. Esposito, D. Di Paola, and I. Matta, "On distributed virtual network embedding with guarantees," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 569–582, 2016.

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

[82] N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. Latré, and F. D. Turck, "Semantically enhanced mapping algorithm for affinity-constrained service function chain requests," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 317–331, June 2017.

[83] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, no. C, pp. 138–155, Nov. 2016.

[84] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, and C. Metz, "Colap: A predictive framework for service function chain placement in a multi-cloud environment," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–9.

[85] Z. Zhang, Z. Li, C. Wu, and C. Huang, "A scalable and distributed approach for nfv service chain cost minimization," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2151–2156.

[86] A. Leivadeas, G. Kesidis, M. Falkner, and I. Lambadaris, "A graph partitioning game theoretical approach for the vnf service chaining problem," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 890–903, Dec 2017.

[87] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[88] B. Kar, E. H. K. Wu, and Y. D. Lin, "Energy cost optimization in dynamic placement of virtualized network function chains," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 372–386, March 2018.

[89] H. Ko, D. Suh, H. Baek, S. Pack, and J. Kwak, "Optimal placement of service function in service function chaining," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2016, pp. 102–105.

[90] C. H. Hsieh, J. W. Chang, C. Chen, and S. H. Lu, "Network-aware service function chaining placement in a data center," in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Oct 2016, pp. 1–6.

[91] M. A. Tahmasbi Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vspace: Vnf simultaneous placement, admission control and embedding," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 542–557, March 2018.

[92] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "Service chain (sc) mapping with multiple sc instances in a wide area network," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[93] ——, "A scalable approach for service chain (sc) mapping with multiple sc instances in a wide-area network," *IEEE Journal on Selected Areas in Communications*, vol. PP, no. 99, pp. 1–1, 2018.

[94] Z. Zhu, H. Lu, J. Li, and X. Jiang, "Service function chain mapping with resource fragmentation avoidance," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[95] A. Shameli-Sendi, Y. Jarraya, M. Pourzandi, and M. Cheriet, "Efficient provisioning of security service function chaining using network security defense patterns," *IEEE Transactions on Services Computing*, vol. 12, no. 4, pp. 534–549, July 2019.

[96] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 554–568, Sept 2017.

[97] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 128–133.

[98] G. Moualla, T. Turletti, M. Bouet, and D. Saucez, "On the necessity of accounting for resiliency in sfc," in *2016 28th International Teletraffic Congress (ITC 28)*, vol. 02, Sept 2016, pp. 13–15.

[99] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543–553, Sept 2017.

[100] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and sfc requests in cloud datacenter," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2018.

[101] H. Huang, P. Li, S. Guo, W. Liang, and K. Wang, "Near-optimal deployment of service chains by exploiting correlations between network functions," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[102] J. J. Kuo, S. H. Shen, H. Y. Kang, D. N. Yang, M. J. Tsai, and W. T. Chen, "Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[103] M. T. Beck and J. F. Botero, "Coordinated allocation of service function chains," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[104] ——, "Scalable and coordinated allocation of service function chains," *Computer Communications*, vol. 102, pp. 78 – 88, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416303577

[105] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions in NFV," *CoRR*, vol. abs/1503.06377, 2015. [Online]. Available: http://arxiv.org/abs/1503.06377

[106] C. Ghribi, M. Mechtri, and D. Zeghlache, "A dynamic programming algorithm for joint vnf placement and chaining," in *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, ser. CAN '16. New York, NY, USA: ACM, 2016, pp. 19–24. [Online]. Available: http://doi.acm.org/10.1145/3010079.3010083

[107] C. Ghribi, M. Mechtri, O. Soualah, and D. Zeghlache, "Sfc provisioning over nfv enabled clouds," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, June 2017, pp. 423–430.

[108] M. Jalalitabar, E. Guler, G. Luo, L. Tian, and X. Cao, "Dependence-aware service function chain design and mapping," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[109] Z. Shaoping, G. Xiujiao, and Y. Hongfang, "Virtual network function instantiation and service function chaining mapping in wide area network," in *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, July 2016, pp. 1–6.

[110] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Network service chaining with efficient network function mapping based on service decompositions," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–5.

[111] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 98–106.

[112] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[113] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct 2014, pp. 7–13.

[114] J. Lee, H. Ko, D. Suh, S. Jang, and S. Pack, "Overload and failure management in service function chaining," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–5.

[115] M. Flittner, J. M. Scheuermann, and R. Bauer, "Chainguard: Controller-independent verification of service function chaining in cloud computing," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–7.

[116] R. A. Eichelberger, T. Ferreto, S. Tandel, and P. A. P. R. Duarte, "Sfc path tracer: A troubleshooting tool for service function chaining," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 568–571.

[117] B. Tschaen, Y. Zhang, T. Benson, S. Banerjee, J. Lee, and J. M. Kang, "Sfc-checker: Checking the correct forwarding behavior of service function chaining," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 134–140.

[118] K. Hwang, Y. Shi, and X. Bai, "Scale-out vs. scale-up techniques for cloud performance and productivity," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014, pp. 763–768.

[119] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1206–1243, 2018.

[120] H. Hawilo, M. Jammal, and A. Shami, "Orchestrating network function virtualization platform: Migration or re-instantiation?" in *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, Sept 2017, pp. 1–6.

[121] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2008–2025, Aug 2017.

[122] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 255–260.

[123] F. Z. Yousaf, C. Goncalves, L. Moreira-Matias, and X. C. Perez, "Rava; resource aware vnf agnostic nfv orchestration method for virtualized networks," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sept 2016, pp. 1–6.

[124] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in nfv: Models and algorithms," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 534–546, July 2017.

[125] M. D. Mauro, M. Longo, F. Postiglione, G. Carullo, and M. Tambasco, "Service function chaining deployed in an nfv environment: An availability modeling," in *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, Sept 2017, pp. 42–47.

[126] J. Nam, J. Seo, and S. Shin, "Probius: Automated approach for vnf and service chain analysis in software-defined nfv," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '18. New York, NY, USA: ACM, 2018, pp. 14:1–14:13. [Online]. Available: http://doi.acm.org/10.1145/3185467.3185495

[127] A. G. Tasiopoulos, S. G. Kulkarni, M. Arumaithurai, I. Psaras, K. K. Ramakrishnan, X. Fu, and G. Pavlou, "Drench: A semi-distributed resource management framework for nfv based service function chaining," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, June 2017, pp. 1–9.

[128] Y. J. Chen, L. C. Wang, F. Y. Lin, and B. S. P. Lin, "Deterministic quality of service guarantee for dynamic service chaining in software defined networking," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 991–1002, Dec 2017.

[129] A. Dwaraki and T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks," in *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMIddlebox '16. New York, NY, USA: ACM, 2016, pp. 32–37. [Online]. Available: http://doi.acm.org/2940147.2940148

[130] M. Gharbaoui, S. Fichera, P. Castoldi, and B. Martini, "Network orchestrator for qos-enabled service function chaining in reliable nfv/sdn infrastructure," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–5.

[131] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, "Software defined network service chaining for ott service providers in 5g networks," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 124–131, NOVEMBER 2017.

[132] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in sdn-enabled networks with middleboxes," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, Nov 2016, pp. 1–10.

[133] G. Lee, M. Kim, S. Choo, S. Pack, and Y. Kim, "Optimal flow distribution in service function chaining," in *The 10th International Conference on Future Internet*, ser. CFI '15. New York, NY, USA: ACM, 2015, pp. 17–20. [Online]. Available: http://doi.acm.org/10.1145/2775088.2775103

[134] B. Yi, X. Wang, and M. Huang, "Design and evaluation of schemes for provisioning service function chain with function scalability," *Journal of Network and Computer Applications*, vol. 93, pp. 197 – 214, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517302102

[135] 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. [Online]. Available: https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-IFA030v3.2.1-GS-MultiDomainMANO-spec.pdf

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

[137] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in nfv networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, Feb 2017.

[138] F. Esposito, "Catena: A distributed architecture for robust service function chain instantiation with guarantees," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, July 2017, pp. 1–9.

[139] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, "Service function chaining in multi-domain networks," in *2016 Optical Fiber Communications Conference (OFC)*, March 2016, pp. 1–3.

[140] A. Abujoda and P. Papadimitriou, "Distnse: Distributed network service embedding across multiple providers," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2016, pp. 1–8.

[141] N. Figueira and R. R. Krishnan, "Sdn multi-domain orchestration and control: Challenges and innovative future directions," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2015, pp. 406–412.

[142] R. Guerzoni et al., "Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey," *Trans. on Emerg. Telecomm. Technol.*, vol. 28, no. 4, p. e3103.

[143] D. Dietrich, A. Abujoda, A. Rizk, and P. Papadimitriou, "Multi-provider service chain embedding with nestor," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 14, no. 1, pp. 91–105, March 2017.

[144] Q. Xu, D. Gao, T. Li, and H. Zhang, "Low latency security function chain embedding across multiple domains," *IEEE Access*, vol. 6, pp. 14 474–14 484, 2018.

[145] G. Sun, Y. Li, D. Liao, and V. Chang, "Service function chain orchestration across multiple domains: A full mesh aggregation approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 1175–1191, Sep. 2018.

[146] M. Shen, K. Xu, K. Yang, and H. H. Chen, "Towards efficient virtual network embedding across multiple network domains," in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*, May 2014, pp. 61–70.

[147] Gurobi, http://www.gurobi.com/.

[148] NetworkX, https://networkx.github.io/.

[149] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, Apr 2004. [Online]. Available: https://doi.org/10.1007/s00158-003-0368-6

[150] A. Messac, "Physical programming - effective optimization for computational design," *AIAA Journal*, vol. 34, no. 1, pp. 149–158, Jan 1996. [Online]. Available: https://doi.org/10.2514/3.13035

[151] A. Messac, S. Gupta, and B. Akbulut, "Linear physical programming: A new approach to multiple objective optimization," *Transactions on Operational Research*, vol. 8, 01 1996.

[152] J. Sanchis, M. A. Martinez, X. Blasco, and G. Reynoso-Meza, "Modelling preferences in multi-objective engineering design," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 8, pp. 1255 – 1264, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197610001429

[153] Z.-G. Tian, H.-Z. Huang, and L.-W. Guan, "Fuzzy physical programming and its application in optimization of through passenger train plan," 07 2002, pp. 498–503.

[154] I. Mehmet Ali and G. Surendra, "Physical programming: A review of the state of the art," *Studies in Informatics and Control*, vol. 21, no. 4, pp. 349–366, 2012.

[155] N. Zhang, "Physical programming based multidisciplinary optimization for aircraft conceptual parameter design," in *2011 Chinese Control and Decision Conference (CCDC)*, May 2011, pp. 2387–2392.

[156] ——, "2d turbine airfoil optimization using physical programming," in *2010 IEEE International Conference on Mechatronics and Automation*, Aug 2010, pp. 852–856.

[157] R. Chai, A. Savvaris, A. Tsourdos, and Y. Xia, "An interactive fuzzy physical programming for solving multiobjective skip entry problem," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 5, pp. 2385–2398, Oct 2017.

[158] H. Li, M. Ma, and W. Zhang, "Multi-objective collaborative optimization using linear physical programming with dynamic weight," *Journal of Mechanical Science and Technology*, vol. 30, no. 2, pp. 763–770, Feb 2016. [Online]. Available: https://doi.org/10.1007/s12206-016-0131-8

[159] S. An, S. Yang, Y. Bai, and X. Wu, "An improved physical programming method for multi-objective inverse problems," in *International Journal of Applied Electromagnetics and Mechanics*, vol. 52, no. 3-4, 2016, pp. 1151–1159.

[160] Y. Shao and J. Yu, "Evaluation of transmission expansion scheme based on physical programming," in *2012 Asia-Pacific Power and Energy Engineering Conference*, March 2012, pp. 1–5.

[161] Linearization of the product of two variables, https://www.leandro-coelho.com/linearization-product-variables/.

[162] X. Ma and B. Dong, "Linear physical programming-based approach for web service selection," in *2008 International Conference on Information Management, Innovation Management and Industrial Engineering*, vol. 2, Dec 2008, pp. 398–401.

[163] CPLEX, https://www.ibm.com/analytics/cplex-optimizer.

[164] J. Alqahtani and B. Hamdaoui, "Rethinking fat-tree topology design for cloud data centers," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[165] K. Solnushkin, https://clusterdesign.org/fat-trees/.

[166] GSMA, "Network Slicing Use Case Requirements," GSMA, White Paper (WP), 04 2018.

[167] 3GPP, "Technical Specification Group Services and System Aspects;Policy and charging control architecture," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.203, 09 2018, version 15.4.0.

[168] I. Parvez, A. Rahmati, I. Güvenç, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *CoRR*, vol. abs/1708.02562, 2017. [Online]. Available: http://arxiv.org/abs/1708.02562

[169] S. Khatibi, "5g mobile network architecture for diverse services, use cases, and applications in 5g and beyond deliverable d6.1 documentation of requirements and kpis and definition of suitable evaluation criteria contractual date of delivery," Tech. Rep., 09 2017.

[170] X. Zhong, Y. Wang, and X. Qiu, "Service function chain orchestration across multiple clouds," *China Communications*, vol. 15, no. 10, pp. 99–116, Oct 2018.

[171] J. Zu, G. Hu, Y. Wu, D. Shao, and J. Yan, "Resource aware chaining and adaptive capacity scaling for service function chains in distributed cloud network," *IEEE Access*, vol. 7, pp. 157 707–157 723, 2019.

[172] J. Pei, P. Hong, K. Xue, and 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*, vol. 30, no. 10, pp. 2179–2192, 2019.

[173] R. V. Rosa, M. A. S. Santos, and C. E. Rothenberg, "Md2-nfv: The case for multi-domain distributed network functions virtualization," in *2015 International Conference and Workshops on Networked Systems (NetSys)*, March 2015, pp. 1–5.

[174] 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 and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, Dec 2016, pp. 189–195.

[175] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, "On multi-domain network slicing orchestration architecture and federated resource control," *IEEE Network*, vol. 33, no. 5, pp. 242–252, 2019.

[176] J. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, and S. Sargento, "Toward a telco cloud environment for service functions," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 98–106, Feb 2015.

[177] S. Kumar, M. Tufail, S. Majee, C. Captari, and S. Homma, "Service Function Chaining Use Cases In Data Centers," Internet Engineering Task Force, Internet-Draft draft-ietf-sfc-dc-use-cases-06, Feb. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sfc-dc-use-cases-06

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

[179] B. Martini, F. Paganelli, A. A. Mohammed, M. Gharbaoui, A. Sgambelluri, and 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)*, April 2015, pp. 1–5.

[180] P. B. Pawar and K. Kataoka, "Segmented proactive flow rule injection for service chaining using sdn," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 38–42.

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

[182] 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.

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

[184] Open Vswitch, http://docs.openvswitch.org/en/latest/intro/what-is-ovs/.

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

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

[187] R. Boutaba, M. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. Caicedo Rendon, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, 05 2018.

[188] Y. Liu, H. Lu, X. Li, D. Zhao, W. Wu, and G. Lu, "A novel approach for service function chain dynamic orchestration in edge clouds," *IEEE Communications Letters*, pp. 1–1, 2020.

[189] J. Pei, P. Hong, and D. Li, "Virtual network function selection and chaining based on deep learning in sdn and nfv-enabled networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[190] J. Pei, P. Hong, K. Xue, D. Li, D. S. L. Wei, and F. Wu, "Two-phase virtual network function selection and chaining algorithm based on deep learning in sdn/nfv-enabled networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1102–1117, 2020.

[191] S. Lange, N. Van Tu, S. Jeong, D. Lee, H. Kim, J. Hong, J. Yoo, and J. W. Hong, "A network intelligence architecture for efficient vnf lifecycle management," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2020.

[192] S. Jeong, H. Kim, J. Yoo, and J. W. Hong, "Machine learning based link state aware service function chaining," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019, pp. 1–4.

[193] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[194] S. I. Kim and H. S. Kim, "Method for vnf placement for service function chaining optimized in the nfv environment," in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, 2019, pp. 721–724.

[195] H. Chai, J. Zhang, Z. Wang, J. Shi, and T. Huang, "A parallel placement approach for service function chain using deep reinforcement learning," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, 2019, pp. 2123–2128.

[196] W. Mao, L. Wang, J. Zhao, and Y. Xu, "Online fault-tolerant vnf chain placement: A deep reinforcement learning approach," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 163–171.

[197] D. M. Manias, M. Jammal, H. Hawilo, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Machine learning for performance-aware virtual network function placement," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[198] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang, and J. Zhang, "Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, 2019, pp. 1–10.

[199] H. A. Shah and L. Zhao, "Multi-agent deep reinforcement learning based virtual resource allocation through network function virtualization in internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[200] S. Troia, R. Alvizu, and G. Maier, "Reinforcement learning for service function chain reconfiguration in nfv-sdn metro-core optical networks," *IEEE Access*, vol. 7, pp. 167 944–167 957, 2019.

[201] N. Jalodia, S. Henna, and A. Davy, "Deep reinforcement learning for topology-aware vnf resource prediction in nfv environments," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2019, pp. 1–5.