QoS and Resource aware Security Orchestration System

Miloud Bagaa*, Tarik Taleb*†, Jorge Bernal Bernabe‡, Antonio Skarmeta‡
* Aalto University, Finland, {miloud.bagaa, tarik.taleb}@aalto.fi
†University of Oulu, 90570 Oulu, Finland.
‡ Department of Communications and Information Engineering, University of Murcia, Spain {jorgebernal, skarmeta}@um.es

***Abstract*—Network Function Virtualization (NFV) and Software Distributed Networking (SDN) technologies play a crucial role in enabling 5G system and beyond. A synergy between these both technologies has been identified for enabling a new concept dubbed service function chains (SFC) that aims to reduce both the capital expenditures (CAPEX) and operating expenses (OPEX). The SFC paradigm considers different constraints and key performance indicators (KPIs), that includes QoS and different resources, for enabling network slice services. However, the large-scale, complexity and security issues brought by these technologies create an extra overhead for ensuring secure network slicing. To cope with these challenges, this paper proposes a cost-efficient optimized SFC management system that enables the creation of SFCs for enabling efficient and secure network slices. The proposed system considers the network and computational resources and current network security levels to ensure trusted deployments. The simulation results demonstrated the efficiency of the proposed solution for achieving its designed objectives. The proposed solution efficiently manages the SFCs by optimizing deployment costs and reducing overall end-to-end delay**

I. Introduction

The recent advances in network softwarization have staggeringly benefited the 5G system and beyond by increasing the network elasticity and flexibility and reducing capital expenditures (CAPEX) operating expenses (OPEX). Network softwarization leverages both network function virtualization (NFV) and software-defined networking (SDN) paradigms to enable the utilization of common-off-the-shelf (COTS) hardware. The latter is characterized by its fast software upgrade rather than costly hardware replacement and hence enlarges different use-cases by exploiting the same device. NFV leverages virtualization techniques to deploy virtual network functions (VNFs) (e.g., router, switches, firewalls, proxies, AAA services, Virtual-HoneyNet) that run on top of general-purpose hardware. Meanwhile, SDN technology dynamically and effectively connects these VNFs by setting-up service function chains (SFCs) belonging to different service providers (SPs).

The fact of sharing the physical resources across SPs makes it more complex to orchestrate, schedule, and chaining the VNFs. The management of SFC is heightened when dealing with SDN/NFV-based IoT networks, where different virtualized and software clouds and edges are involved. The heterogeneity of these players imposes challenges to manage and optimize the overall end-to-end delay as different heterogeneous virtual network infrastructures in the SFC are required.

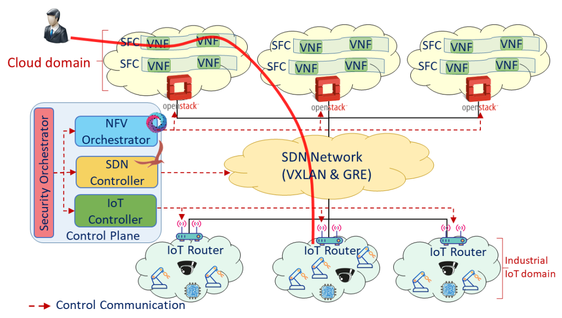
To deal with the above challenges, in this paper, we suggest a cost-efficient SFC deployment and management framework that aims to ensure the desired KPIs that include the QoS and security while reducing the overall cost. The proposed mechanism considers the implementation and management of SFCs that are potentially deployed across IoT domains, inter-clouds, and cross-network segments. Therefore, ensuring efficient management of SFC in a virtualized and software IoT network is a challenging process that is targetted by this paper. To this aim, unlike the current state of the art, besides the QoS and actual capacities of VNFs in terms of resources (CPU, RAM, and storage), the proposed solution also considers the network security levels (enforced security channels mechanisms across the VNFs in the SFC). The algorithm finds a predefined blueprint configuration for the SFC, and targeted KPIs, and optimizes the targeted KPIs in terms of link security level, QoS, delay, and bandwidth.

The remaining of the paper is organized as follows. Section II summarizes the related works. Section III introduces the main idea and the problem formulation targeted by this paper. The optimization algorithm is described in section IV, while section V shows the simulation results and its discussion. Finally, section VI concludes the article.

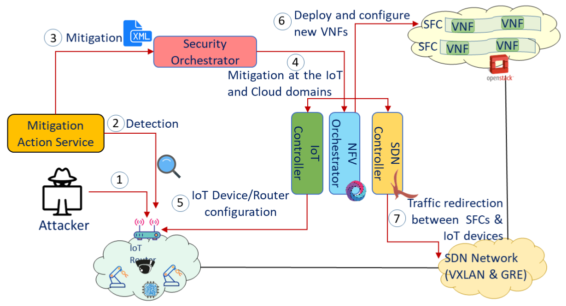
II. Related work

The security is one of the main challenges facing the management and orchestration of SFC [1], [2], and VNFs. The deployment and management of SFC in a dynamic environment is a highly challenging process that attracted the attention of industry and academia in the past. Authors in [3] have suggested a basic formulation of VNF scheduling that considers the processing delay at different VNFs in the chain. Meanwhile, authors in [4] bring up simple algorithms that aim at minimizing the time, the cost, and the revenue in function mapping and scheduling.

Authors in [5] have studied the problem of VNFs allocation and deployment in a distributed environment. However, the authors do not consider the placement of these VNFs in different SFCs. Similarly, authors in [6] have addressed the service chaining optimization challenge. However, they do not find the end-to-end delay when deploying the SFCs. Similarly, in [7], authors formalize the VNF placement and SFC problem as an Integer Linear Programming (ILP). They solve the optimization problem by considering the cost and the distance across clients and VNFs. However, the proposed optimization does not find other resources and security aspects, as addressed in our model. Moreover, in contrast to this solution, our solution considers cost-efficient life cycle management when allocating computational and network resources. Some other related research papers mainly focus on different kinds of QoS metrics for improving performances, such as bandwidth allocation optimization for VNFs usage [8]. Meanwhile, authors [9] authors analyze the VNF scheduling and resource allocation



(a) Main Architecture Overview


6 Deploy and configure new VNFs
3 Mitigation
XML
Security Orchestrator
4 Mitigation at the IoT and Cloud domains
SFC
VNF
VNF
SFC
VNF
VNF
openstack
Mitigation Action Service
2 Detection
IoT Controller
NFV Orchestrator
SDN Controller
1
Attacker
5 IoT Device/Router configuration
Traffic redirection
7 between SFCs & IoT devices
IoT Router
SDN Network (VXLAN & GRE)

(b) Detection and Mitigation of Different Threats

Fig. 1: Envisioned System: Architecture and Security Enablers

with SFCs and provide resource optimization solutions. They define the scheduling decisions and chaining based on service transmission and VNF's processing delays. However, unlike the proposed algorithm, they do not consider security aspects in the formulation.

Authors in [10] have suggested a reliability-aware SFC solution that aims to provision VNFs with delay guarantees. To solve the problem, they have suggested the use of a mixed-integer linear program (MILP) for VNFs allocation that maximizes reliability and end-to-end delays. From another side, Artificial Intelligence has been leveraged in [11] for ensuring efficient VNF scheduling. Unfortunately, the work considers only the processing and transmission delay when scheduling the SFCs. Meanwhile, Long Qu et al. [12] have proposed a model for SFC orchestration in 5G networks. The solution has been modeled as ILP, and then heuristic algorithms have been suggested. However, the proposed solutions consider only the minimization of the path length and cost of computing resources.

III. Network model and problem formulation

A. Main idea

Fig. 1 depicts the envisioned system targeted by this paper. As illustrated in Fig. 1, the user can access to IoT domains via SFCs that are deployed on multiple administrative and technological domains. The management of these SFCs happens at the security orchestrator (SO). The latter can manage the connection and mitigate the attacks by leveraging: $i)$ NFV orchestrator; $ii)$ SDN controller; $iii)$ IoT controller. In the the suggested framework, when an attack is detected by the mitigation action service (MAS) component, a mitigation request (security policy) should be generated and sent to the security orchestrator (SO) (Fig. 1(b), Steps $1-3$). The latter should run an optimization algorithm, suggested in this paper, to mitigate the attacks by deploying or redirecting traffic through existing SFCs by taking into account the actual resources, the requested QoS, and security aspects.

Furthermore, the new SFCs can use either complete new security VNF instances (VNFIs) or already existing ones. When designing and deploying new SFCs, the SO considers the used security VNFs and their communication links for ensuring the desired security and QoS. For instance, the communication between different clouds, VNFs, users, and devices, is characterized by unusual delay, bandwidth, jitters, and security levels. For instance, the communication link within the same cloud has a high-security level, while the one between users, devices, and inter clouds would have more vulnerability. Moreover, the link security levels can be affected by using security mechanisms, such as IPsec, SSL or even DTLs when targeting IoT environments. Later, the SO will mitigate the attacks by communicating with IoT controllers, NFV orchestrators, and/or SDN controllers (Fig. 1(b), Step 4).

The attacks would be mitigated at the edge by communicating with the IoT controller. For instance, the malicious IoT devices could be reconfigured (using protocols such as CoAP or LWM2M), shut-off, rebooted and even flushed, thereby preventing the attacks to be spread from a domain to another (Fig. 1(b), Step 5). The attacks could also be mitigated via the SDN controller by either preventing the attacks, e.g. adding new flow rules to drop the traffic from the attacker or limiting the bandwidth (Fig. 1(b), Step 6). Finally, the attacks can also be mitigated by deploying or redirecting traffic through SFCs (Fig. 1(b), Steps $6-7$).

TABLE I: Summary of Notations.

Notation	Description
$\mathcal{U}$	The set of users in the network.
$\mathcal{D}$	The IoT devices deployed in a distributed IoT infrastructure.
V_C	The set of clouds or edges existing in the network.
$\mathcal{R}$	The set of all the resources. $\mathcal{R}$ could equal to $\{\text{RAM}, \text{CPU}, \text{DISK}\}$.
$\Delta_r(c)$	A parameter that shows the amount of resource $r \in \mathcal{R}$ available at the cloud/edge $c \in V_C$.
Θ	The set of service function chaining (SFCs) that serves the users.
Φ_θ	The users of the SFC $\theta \in \Theta$.
λ_θ	The expected traffic that will be exchanged between the users Φ_θ and the IoT devices $\mathcal{D}$.
Υ_θ	The list of VNFs forming the SFC $\theta \in \Theta$.
Π	The set of VNFs' types supported in the system, such as firewall, load balancer,...etc.
π_υ	The type of the VNF $\upsilon \in \Upsilon_\theta$, such that $\pi_\upsilon \in \Pi$.
$\mathcal{V}$	The VNF instances that should be instantiated to serve the VNFs $\bigcup\limits_{\theta \in \Theta} \Upsilon_\theta$.
$\zeta(\upsilon)$	All the VNFs that have the same type like the VNF $\upsilon \in \bigcup\limits_{\theta \in \Theta} \Upsilon_\theta$. $\zeta(\upsilon)$ is defined formally as follow: $\zeta(\upsilon) = \{\acute{\upsilon} : \forall \acute{\upsilon} \in \bigcup\limits_{\acute{\theta} \in \Theta} \Upsilon_{\acute{\theta}} \wedge \pi_\theta = \pi_{\acute{\theta}}\}$.
$\Psi(\upsilon)$	All the VNFs that have conflicts with the VNF $\upsilon \in \Upsilon_\theta$.
$\mathcal{F}$	The list of flavors that can be used by different VNFs. Each flavor $f \in \mathcal{F}$ specifies the amount of all resources $r \in \mathcal{R}$ that should be used by a VNF instance.
$\delta_r(f)$	The amount of resources $r \in \mathcal{R}$ used by the flavor $f \in \mathcal{F}$.
$\delta_p(f)$	The price of using the flavor $f \in \mathcal{F}$.
$\Omega(u)$	The set of VNFs that would be hosted at the VNFI $u \in \mathcal{V}$.
$\Gamma(.)$	$\Gamma(\pi_u, f, \sum\limits_{\theta \in \Theta, \upsilon \in \Upsilon_\theta \cap \Omega(\upsilon)} \lambda_\theta)$ denotes the average processing delay would be expected from the VNFI u to process one packet.

B. Problem formulation

We denote by $\mathcal{U}$ a set of users that interested in different services offered by IoT devices $\mathcal{D}$ deployed in a distributed IoT infrastructure. Let $\mathcal{G}(V, E, W)$ denote the underlying network that consists of a set of IoT devices and clouds. V consists of: $i)$ Clouds or edges (V_C); $ii)$ Distributed IoT infrastructure (V_I); $iii)$ End-users (V_U) that represent the (radio) access nodes ((R)AN) from whereby the users access. Each cloud/edge $u \in V_C$ is characterized by limited storage and computation resources, including CPU, RAM, and DISK. Let $\Delta_r(u)$ be a vector that shows the resources of the cloud $u \in V_C$. For the sake of simplicity, E represents the end-to-end connections between $V_U \cup V_I \cup V_C$ by considering an overlay network and by making an abstraction on the intermediate routers and switches (backbone). However, the suggested model is orthogonal and can be easily adapted and extended to consider the intermediate network components if needed. Meanwhile, W denotes the characteristics of the links that include the bandwidth, delay, and security level. Let $\mathcal{W}^{\mathcal{B}}$, $\mathcal{W}^{\mathcal{L}}$ and $\mathcal{W}^{\mathcal{S}}$ denote the characteristics of the links $\mathcal{E}$ in terms of bandwidth capacity, propagation delay and security level, respectively.

We denote by Θ the set of SFCs in the network, such that each SFC $\theta \in \Theta$ serves a set of users. Let Φ_θ denote the set of users of the SFC $\theta \in \Theta$. The SO would have the information about the expected traffic that would be exchanged between the users Φ_θ and the IoT devices. Let λ_θ denote the expected traffic that will be exchanged between the users Φ_θ and the IoT devices.

In the proposed model, we leverage the strength of SDN technology to interconnect the users with their IoT devices. Formally, the communication between a user and its IoT devices happens through a specified SFC $\theta \in \bar{\Theta}$ that consists of a list of VNFs Υ_θ. We denote by Π the set of VNFs types supported by the system. We denote by π_v the type of the VNF $v \in \Upsilon_\theta$, such that $\pi_v \in \Pi$. We denote by $(v_i, v_j) \in \Upsilon_\theta$ a two consecutive VNFs in the SFC $\theta \in \bar{\Theta}$. In the blueprint, the KPIs of each SFC $\theta \in \bar{\Theta}$ consider the end-to-delay and bandwidth, as well as the security levels of the links used for interconnecting θ's components. While the two former metrics target the quality of experience (QoE), the latter targets the security level of θ. Let $\sigma_\theta^{\mathcal{L}}$, $\sigma_\theta^{\mathcal{B}}$ and $\sigma_\theta^{\mathcal{S}}$, denote the θ's end-to-end delay, bandwidth and security level, respectively. Let $\mathcal{V}$ represent the instantiated VNFs in different clouds. $\mathcal{V}$ represents the VNF instances (VNFI) that would be used by the VNFs Θ. Note that more than one VNF in $(\bar{\Theta})$ could use the same VNFI in $(\mathcal{V})$. Based on the observation that each new VNF would be maximum instantiated at one VNFI, we have $|\mathcal{V}| \leq |\bigcup_{\theta \in \Theta} \Upsilon_\theta|$.

In the proposed model, one VNFI can be shared with multiple SFCs, and hence two VNFs $(v_1, v_2) \in \Upsilon_{\theta_1} \times \Upsilon_{\theta_2}$, that belong to two different SFCs $\theta_1, \theta_2 \in \bar{\Theta}$, could use the same VNFI $u \in \mathcal{V}$ if and only if the following conditions hold: $i)$ They have the same type $\pi_{v_1} = \pi_{v_2}$; $ii)$ There are no conflicts in the configuration of the two VNFs v_1 and v_2. Let $\zeta(v)$ denote all the VNFs that have the same type like v. Formally $\zeta(v)$ is defined as follows: $\zeta(v) = \{\acute{v} : \forall \acute{v} \in \bigcup_{\acute{\theta} \in \Theta} \Upsilon_{\acute{\theta}} \wedge \pi_v = \pi_{\acute{v}}\}$. Meanwhile, $\Psi(v)$ denotes all the VNFs that have conflicts with the VNF $v \in \Upsilon_\theta$. The SO receives this list from the "Policy Interpreter" component. Formally, the candidate list of VNFs that can be hosted in the same VNFI with the VNF $v \in \Upsilon_\theta$ is $\zeta(v) - \Psi(v)$. Let $\mathcal{R}$ denote the resources that are considered. $\mathcal{R}$ can be defined as follows: $\mathcal{R} = \{\text{RAM}, \text{CPU}, \text{STORAGE}\}$. Also, we denote by $\mathcal{F}$ the list of available flavors in the network. Each flavor $f \in \mathcal{F}$ has a specified amount of resources should be used by a VNF instance. Let $\delta_r(f)$ denote the amount of resources $r \in \mathcal{R}$ should be used by the favor f. We denote by $\Omega(u)$ the set of VNFs that would be hosted at the VNFI $u \in \mathcal{V}$. Formally, there is a correlation between the resources used by a VNFI $u \in \mathcal{V}$, the amount of traffic treated by that VNFI, and the expected services offered by that VNFI in terms of computation and QoS. While the amount of traffic negatively affects the QoS, the amount of resources has a positive impact on the QoS [13], [14].

Let π_u denote the VNF instance type. Note that a VNFI u whose type is similar to the VNFs that host. Let $\Gamma(\pi_u, f, \sum_{\theta \in \Theta, v \in \Upsilon_\theta \cap \Omega(v)} \lambda_\theta)$ denote a function that returns the average expected processing delay for handling the data traffic $\sum_{\theta \in \Theta, v \in \Upsilon_\theta \cap \Omega(v)} \lambda_\theta$ by a VNFI u that uses the flavor $f \in \mathcal{F}$ and has a type $\pi_u \in \Pi$. For the sake of simplicity, we consider that $\Gamma(.)$ is a linear function in respect to the amount of data λ_θ.

IV. Proposed solution

In this section, we present the solution that optimizes the deployment of SFCs by taking into account the QoS and the security. In this section, we first start by defining the different variables, and then we will describe the various constraints.

A. VNF and VNFI relationship constraints

We define by $\mathcal{X}_{v,u}$ a decision Boolean variable that shows if the VNF $v \in \Upsilon_\theta$ of the SFC $\theta \in \Theta$ will use the VNFI $u \in \mathcal{V}$. Formally, $\mathcal{X}_{v,u} = 1$ if v uses VNFI u, otherwise $\mathcal{X}_{v,u} = 0$. $\mathcal{X}_{v,u}$ should be subject to the following constraint. VNFs in each SFC θ should use only one VNFI. Each VNF, $v \in \Upsilon_\theta$ for $\theta \in \hat{\Theta}$, should use only and only one VNFI u in the network.

$$\forall \theta \in \Theta, \forall v \in \Upsilon_\theta : \sum_{u \in \mathcal{V}} \mathcal{X}_{v,u} = 1 \tag{1}$$

From another side, let $\mathcal{B}_{\theta,u}$ denote a Boolean decision variable that shows if an SFC $\theta \in \hat{\Theta}$ would use VNFI $u \in \mathcal{V}$. Formally, $\mathcal{B}_{\theta,u} = 1$ if u is used by θ, otherwise $\mathcal{B}_{\theta,u} = 0$. For preventing the loop in the SFC θ, VNFs ($\forall v \in \Upsilon_\theta$) of that SFC should use the VNFI u maximum once as mentioned by constraint (2).

$$\forall u \in \mathcal{V}, \forall \theta \in \Theta : \sum_{v \in \Upsilon_\theta} \mathcal{X}_{v,u} = \mathcal{B}_{\theta,u} \tag{2}$$

The following constraints ensure that a VNF v should not share a VNFI $u \in \mathcal{V}$ with another VNF with whom has a conflict or does not have the same type.

We define by $\mathcal{Y}_{v,c}$ a decision Boolean variable that shows if a VNF $v \in \Upsilon_\theta$ for $\theta \in \Theta$ is hosted at the cloud $c \in V_C$ or not. $\mathcal{Y}_{v,c}$ equals to 1 if the VNF v is hosted in the cloud c, otherwise $\mathcal{Y}_{v,c} = 0$. Formally a VNF v is hosted at a cloud $c \in V_C$ if its VNFI is running on top of that cloud. Each VNF should run on top only one cloud at a given time. Constraint (3) ensures that each VNF should be deployed only on one cloud.

$$\forall \theta \in \Theta, \forall v \in \Upsilon_\theta : \sum_{c \in V_C} \mathcal{Y}_{v,c} = 1 \tag{3}$$

We define also by $\mathcal{A}_{u,\pi}$ a decision Boolean variable that shows if the VNFI $u \in \hat{\mathcal{V}}$ has the type $\pi \in \Pi$. Formally, $\mathcal{A}_{u,\pi} = 1$ if and only if VNFI u has the type π. Each VNF $\upsilon \in \Upsilon_\theta$ for $\forall\theta \in \Theta$ that uses VNFI $u \in \mathcal{V}$ should use the same VNF type as mentioned in constraint 4.

$$\forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta, \forall u \in \mathcal{V}, \forall\pi \in \Pi : \pi_\upsilon \times \mathcal{X}_{\upsilon,u} = \pi \times \mathcal{A}_{u,\pi} \times \mathcal{X}_{\upsilon,u} \tag{4}$$

Unfortunately, the constraint (4) is not linear due to the part $\mathcal{A}_{u,\pi} \times \mathcal{X}_{\upsilon,u}$. In order to make this constraint linear, we replace the constraint (4) with the constraints (5) and (6).

$$\forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta, \forall u \in \mathcal{V}, \forall\pi \in \Pi : \pi_\upsilon \leq \pi + (2 - \mathcal{X}_{\upsilon,u} - \mathcal{A}_{u,\pi}) \times \mathcal{M} \tag{5}$$

, where $\mathcal{M}$ is a big number ($\mathcal{M} \approx \infty$).

$$\forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta, \forall u \in \mathcal{V}, \forall\pi \in \Pi : \pi \leq \pi_\upsilon + (2 - \mathcal{X}_{\upsilon,u} - \mathcal{A}_{u,\pi}) \times \mathcal{M} \tag{6}$$

The following constraint ensures that each VNFI $u \in \mathcal{V}$ should have only one type if and only of it is deployed, otherwise it should not have any type.

$$\forall u \in \mathcal{V} : \sum_{\pi\in\Pi} \mathcal{A}_{u,\pi} = \sum_{c\in V_C} \mathcal{U}_{u,c} \tag{7}$$

Constraints (5), (6) and (7) ensure VNFs should have the same type of their VNFIs, which prevent the deployment of the VNFs that have different types on the same VNFI.

B. VNFI and cloud relationship constraints

We also define by $\mathcal{U}_{u,c}$ a Boolean variable that shows if the VNFI $u \in \mathcal{V}$ would be deployed in the cloud/edge $c \in V_C$. Formally, $\mathcal{U}_{u,c} = 1$ if and only if u will be deployed in the cloud c, otherwise $\mathcal{U}_{u,c} = 0$. First of all, a VNFI $u \in \mathcal{V}$ should be deployed if and only if it is used at least by one VNF as presented by the constraint (8).

$$\forall u \in \mathcal{V}, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \sum_{c\in V_C} \mathcal{U}_{u,c} \geq \mathcal{X}_{\upsilon,u} \tag{8}$$

Also, a VNFI $u \in \mathcal{V}$ should be not deployed if it is not used by any VNF in the network as shown in the constraint (9).

$$\forall u \in \mathcal{V} : \sum_{c\in V_C} \mathcal{U}_{u,c} \leq \sum_{\theta\in\Theta,\upsilon\in\Upsilon_\theta} \mathcal{X}_{\upsilon,u} \tag{9}$$

Finally, each VNFI $u \in \mathcal{V}$ should be deployed at most at one cloud as shown in the constraint (10).

$$\forall u \in \mathcal{V} : \sum_{c\in V_C} \mathcal{U}_{u,c} \leq 1 \tag{10}$$

A VNF should use the same cloud where its VNFI is deployed. Constraint (11) ensures that each VNF should use the same cloud as its VNFI. While constraint (12) ensures that a VNF υ should not use a cloud c if its VNFI is not running on that cloud.

$$\forall c \in V_C, \forall u \in \mathcal{V}, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \mathcal{Y}_{\upsilon,c} \geq \mathcal{X}_{\upsilon,u} \times \mathcal{U}_{u,c} \tag{11}$$

$$\forall c \in V_C, \forall u \in \mathcal{V}, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \mathcal{Y}_{\upsilon,c} \leq \sum_{u\in\mathcal{V}} \mathcal{X}_{\upsilon,u} \times \mathcal{U}_{u,c} \tag{12}$$

Unfortunately, the constraint (11) and (12) are not linear. To make the optimization linear, we have replaced the constraints (11) and (12) with the following variables and constraints. First we define the variable $\mathcal{Y}_{\upsilon,u,c}$ for $\upsilon \in \Upsilon_\theta$, $u \in \mathcal{V}$, and $c \in V_C$, respectively. Formally, $\mathcal{Y}_{\upsilon,u,c} = 1$ if and only if VNF υ uses VNFI u that is deployed in the cloud/edge c. Otherwise, $\mathcal{Y}_{\upsilon,u,c} = 0$. Then, we replace the constraint (11) and (12) with the following constraints:

Constraints (13) and (14) ensure that $\mathcal{Y}_{\upsilon,u,c} = 0$ if υ does not use u (i.e., $\mathcal{X}_{\upsilon,u} = 0$) or u is not deployed at cloud c (i.e., $\mathcal{U}_{u,c} = 0$).

$$\forall c \in V_C, \forall u \in \mathcal{V}, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \mathcal{Y}_{\upsilon,u,c} \leq \mathcal{U}_{u,c} \tag{13}$$

$$\forall c \in V_C, \forall u \in \mathcal{V}, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \mathcal{Y}_{\upsilon,u,c} \leq \mathcal{X}_{\upsilon,u} \tag{14}$$

Constraint (15) ensures that $\mathcal{Y}_{\upsilon,u,c} = 1$ if and only if both variables $\mathcal{X}_{\upsilon,u}$ and $\mathcal{U}_{u,c}$ equal to 1.

$$\forall c \in V_C, \forall u \in \mathcal{V}, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \mathcal{Y}_{\upsilon,u,c} \geq \mathcal{X}_{\upsilon,u} + \mathcal{U}_{u,c} - 1 \tag{15}$$

Meanwhile, constraint (16) ensures that the VNF υ is deployed in cloud c (i.e., $\mathcal{Y}_{\upsilon,c} = 1$) if $\mathcal{Y}_{\upsilon,u,c} = 1$.

$$\forall c \in V_C, \forall u \in \mathcal{V}, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \mathcal{Y}_{\upsilon,c} \geq \mathcal{Y}_{\upsilon,u,c} \tag{16}$$

Constraint (17) ensures that a VNF υ is not deployed in a cloud c (i.e., $\mathcal{Y}_{\upsilon,c} = 0$) if it does not use any VNFI deployed at cloud c.

$$\forall c \in V_C, \forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \mathcal{Y}_{\upsilon,c} \leq \sum_{u\in\mathcal{V}} \mathcal{Y}_{\upsilon,u,c} \tag{17}$$

Meanwhile, the constraint (18) ensures that each VNF υ should be hosted at a cloud.

$$\forall\theta \in \Theta, \forall\upsilon \in \Upsilon_\theta : \sum_{c\in V_C} \mathcal{Y}_{\upsilon,c} = 1 \tag{18}$$

C. Resource aware constraints

We define by $\phi_{u,f}$ a decision Boolean variable that shows if VNFI $u \in \mathcal{V}$ would use a flavor $f \in \mathcal{F}$. Formally, $\phi_{u,f} = 1$ if u uses flavor f, otherwise $\phi_{u,f} = 0$. First of all, each VNFI should use only one flavor if it is deployed as shown in the constraint (19).

$$\forall u \in \mathcal{V} : \sum_{f\in\mathcal{F}} \phi_{u,f} = \sum_{c\in V_C} \mathcal{U}_{u,c} \tag{19}$$

The constraint (20) ensures that each cloud/edge should not overloaded. The amount of resource $r \in \mathcal{R}$ should not exceed the capacity $\Delta_r(c)$ of the cloud $c \in V_C$ is not overloaded.

$$\forall r \in \mathcal{R}, \forall c \in V_C : \sum_{f\in\mathcal{F}} \sum_{u\in\mathcal{V}} \delta_r(f) \times \phi_{u,f} \times \mathcal{U}_{u,c} \leq \Delta_r(c) \tag{20}$$

Unfortunately, the constraint (20) is not linear due to the part $\phi_{u,f} \times \mathcal{Y}_{u,c}$. To convert the optimization to linear optimization. we replace the constraint (20) by the following constraints and variables:

First, we define the variable $\mathcal{U}_{u,c,f}$ that shows if the VNFI u uses the flavor f in the cloud c. Formally, $\mathcal{U}_{u,c,f} = 1$ if and only if u uses flavor f in cloud c. Also, we replace the constraint 20 with the following constraints.

$$\forall c \in V_C, \forall f \in \mathcal{F}, u \in \mathcal{V} : \mathcal{U}_{u,c,f} \leq \mathcal{U}_{u,c} \tag{21}$$

$$\forall c \in V_C, \forall f \in \mathcal{F}, u \in \mathcal{V} : \mathcal{U}_{u,c,f} \leq \phi_{u,f} \tag{22}$$

$$\forall c \in V_C, \forall f \in \mathcal{F}, u \in \mathcal{V} : \mathcal{U}_{u,c,f} \geq \mathcal{U}_{u,c} + \phi_{u,f} - 1 \tag{23}$$

$$\forall r \in \mathcal{R}, \forall c \in V_C : \sum_{f \in \mathcal{F}} \sum_{u \in \mathcal{V}} \delta_r(f) \times \mathcal{U}_{u,c,f} \leq \Delta_r(c) \tag{24}$$

D. QoS aware constraints

The following constraint ensures that the end-to-end delay in the SFCs is respected. In our model, we consider that the propagation delay dominates the computation one, and hence we ignore the processing delay. Let we define by $\mathcal{T}^{\mathcal{C}}_{c_1,c_2}$ the expected propagation delay between two clouds $c_1, c_2 \in V_C, c_1 \neq c_2$ after binding and deploying the SFCs.

$$\forall c_1, c_2 \in V_C, c_1 \neq c_2 : \mathcal{T}^{\mathcal{C}}_{c_1,c_2} = \frac{1}{\mathcal{W}_{c_1,c_2}} \sum_{\theta \in \Theta} \sum_{(v_1,v_2) \in \Upsilon_\theta} \lambda_\theta \times \mathcal{Y}_{v_1,c_1} \times \mathcal{Y}_{v_2,c_2} \tag{25}$$

However, the constraint (25) is not linear due to the part $\mathcal{Y}_{v_1,c_1} \times \mathcal{Y}_{v_2,c_2}$. In order to make the optimization linear, we replace the constraint (25) by the following variables and constraints. First of all, we add $\mathcal{Y}_{v_1,c_1,v_2,c_2}$ variable that equals to 1 if and only if v_1 and v_2 are hosted at c_1 and c_2, respectively. Otherwise, $\mathcal{Y}_{v_1,c_1,v_2,c_2} = 0$.

$$\forall \theta \in \Theta, \forall (v_1, v_2) \in \Upsilon_\theta, \forall c_1, c_2 \in V_C, c_1 \neq c_2 : \mathcal{Y}_{v_1,c_1,v_2,c_2} \leq \mathcal{Y}_{v_1,c_1} \tag{26}$$

$$\forall \theta \in \Theta, \forall (v_1, v_2) \in \Upsilon_\theta, \forall c_1, c_2 \in V_C, c_1 \neq c_2 : \mathcal{Y}_{v_1,c_1,v_2,c_2} \leq \mathcal{Y}_{v_2,c_2} \tag{27}$$

$$\forall \theta \in \Theta, \forall (v_1, v_2) \in \Upsilon_\theta, \forall c_1, c_2 \in V_C, c_1 \neq c_2 : \mathcal{Y}_{v_1,c_1,v_2,c_2} \geq \mathcal{Y}_{v_1,c_1} + \mathcal{Y}_{v_2,c_2} - 1 \tag{28}$$

$$\forall c_1, c_2 \in V_C, c_1 \neq c_2 : \mathcal{T}^{\mathcal{C}}_{c_1,c_2} = \frac{1}{\mathcal{W}_{c_1,c_2}} \sum_{\theta \in \Theta} \sum_{(v_1,v_2) \in \Upsilon_\theta} \lambda_\theta \times \mathcal{Y}_{v_1,c_1,v_2,c_2} \tag{29}$$

Then, we will define the propagation delay between two consecutive VNFs in an SFC. Let $\mathcal{T}^{V}_{v_1,v_2}$ denote the propagation delay between two consecutive VNFs $(v_1, v_2) \in \Upsilon_\theta$ for $\theta \in \Theta$. Let we define $\mathcal{T}^{V}_{v_1,v_2}$ for SFCs $\theta \in \bar{\Theta}$.

$$\forall c_1, c_2 \in V_C, c_1 \neq c_2, \forall \theta \in \Theta, \forall (v_1, v_2) \in \Upsilon_\theta : \mathcal{T}^{V}_{v_1,v_2} \leq \mathcal{T}^{\mathcal{C}}_{c_1,c_2} + (1 - \mathcal{Y}_{v_1,c_1,v_2,c_2}) \times \mathcal{M} \tag{30}$$

$$\forall c_1, c_2 \in V_C, c_1 \neq c_2, \forall \theta \in \Theta, \forall (v_1, v_2) \in \Upsilon_\theta : \mathcal{T}^{\mathcal{C}}_{c_1,c_2} \leq \mathcal{T}^{V}_{v_1,v_2} + (1 - \mathcal{Y}_{v_1,c_1,v_2,c_2}) \times \mathcal{M} \tag{31}$$

In what follow, we will define the constraints that ensure the end-to-end delay of each SFC is respected. Thus, the end-to-end delay of each SFC $\theta \in \bar{\Theta}$ does not exceed its threshold $\xi^{\mathcal{L}}_\theta$.

$$\forall \theta \in \Theta : \sum_{(v_1,v_2) \in \Upsilon_\theta} \mathcal{T}^{V}_{v_1,v_2} \leq \xi^{\mathcal{L}}_\theta \tag{32}$$

Finally, we need to ensure that all the paths selected by the VNFs of a given SFC respect the constraints of security. Indeed, all the links that interconnect VNFs of the same SFC $\theta \in \Theta$ should have a security level higher than the security level $\xi^{\mathcal{S}}_\theta$.

$$\forall c_1, c_2 \in V_C, c_1 \neq c_2, \forall \theta \in \Theta, \forall (v_1, v_2) \in \Upsilon_\theta : \xi^{\mathcal{X}}_\theta \times \mathcal{Y}_{v_1,c_1,v_2,c_2} \leq \mathcal{W}^{\mathcal{S}}_{c_1,c_2} \tag{33}$$

E. Final Optimization problem

In what follow, we present the global optimization solution that aims to reduce the cost while taking into account the QoS during the deployment of different SFCs.

$$\mathbf{min} \sum_{u \in \hat{\mathcal{V}}, f \in \mathcal{F}} \delta_p(f) \times \phi_{u,f} \tag{34}$$

S.t,

VNF and VNFI relationship constraints:

(1), (2), (3), (5), (6) and (7).

VNFI and cloud relationship constraints:

(8), (9), (10), (13), (14), (15), (16), (17) and (18).

Resource aware constraints:

(19), (21), (22), (23) and (24)

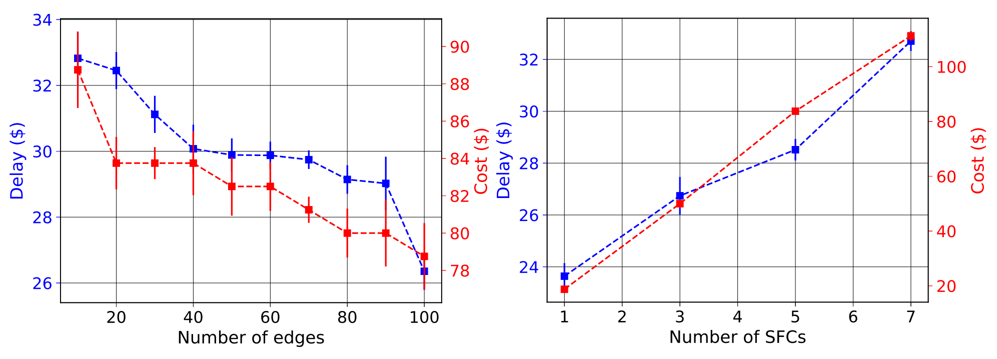
QoS aware constraints:

(26), (27), (28), (29), (30), (31), (32) and (33).

V. Performance evaluation

The proposed solution has been implemented and evaluated using Python language and Gurobi Optimizer software. In all the simulations, we have considered 15 security levels, where we have uniformly distributed them among the connection links that interconnect clouds, edges, and IoT domains. We have evaluated the suggested solution in different scenarios by running 35 repetitions. The plotted results present the mean and 95% confidence interval. First, the average cost that is needed for deploying VNFIs in public clouds for hosting the VNFs of different SFCs. Second, the average end-to-end delay perceived by different SFCs in different epochs.

The suggested solution is evaluated by varying the number of edges and the number of deployed SFCs. In the first scenario, we have varied the number of edges/clouds while fixing the number of SFCs by 4. Meanwhile, in the second scenario, we have modified the number of SFCs while fixing the number of clouds/edges by 40.



(a) Varying the numbers of clouds

(b) Varying the numbers of SFCs

Fig. 2: Cost and Average Delay

Fig. 2(a) depicts the impact of the number of edges on the cost and delay. While the delay is shown with a blue curve with the y-axis in the left, the cost is depicted with a red color curve and y-axis in the right. The first observation that we can draw from this figure is that the number of edges has a positive impact on the cost. Increasing the number of edges/clouds gives more flexibility to the solution for finding a better position for deploying the VNFIs that serve more VNFs, and hence helping more SFCs. This leads to reducing the number of VNFIs that should be used, and therefore reduce the cost. Moreover, increasing the number of edges/clouds also has a positive impact on the delay. Increasing the number of edges/clouds leads to increasing the likelihood of finding better locations for deploying different VNFIs that could offer better end-to-end delay for different SFCs.

Meanwhile, Fig. 2(b) shows the impact of the number of SFCs on the cost and delay, respectively. Increasing the number of SFCs that should be deployed leads to an increase in the number of deployed VNFs, and hence it harms the cost. We also observed that the rise in the number of SFCs hurts the end-to-end delay. This can be explained as follow: Increasing the number of SFCs leads to an increase in the probability of getting VNFs that have more resources' requirements. The resources demanding of these VNFs will limit their options to be deployed in clouds/edges that offer better delays. Also, increasing the number of SFCs leads to raising the likelihood to get SFCs with sparsely distributed users, which hurts the end-to-end delay.

VI. Conclusion

This paper has proposed an efficient Algorithm for deploying different SFC while considering the QoS, actual capacities of VNFs in terms of resources (CPU, RAM, and storage) and current network security levels. The Algorithm has been evaluated via simulation. The obtained results demonstrate the effectiveness of the Algorithm for achieving the desired objectives. As future work we envisage to extend the Algorithm to consider SFC management also in volatile and mobile edge nodes that imposes additional physical constrains to the optimization.

Acknowledgment

This work was partially supported by the European research project H2020 INSPIRE-5Gplus GA 871808. It has been also partially funded by AXA Postdoctoral Scholarship awarded by the AXA Research Fund (Cyber-SecIoT project). This work was partially supported by the Academy of Finland 6Genesis project under Grant No. 318927, and by the Academy of Finland CSN project under Grant No. 311654.

References

[1] N. F. S. de Sousa, D. A. L. Perez, R. V. Rosa, M. A. Santos, and C. E. Rothenberg, "Network service orchestration: A survey," *Computer Communications*, 2019.

[2] L. M. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal-Bernabe, S. N. Srirama, and M. F. Zhani, "Research challenges in nextgen service orchestration," *Future Generation Computer Systems*, vol. 90, pp. 20–38, 2019.

[3] J. F. Riera, E. Escalona, J. Batallé, E. Grasa, and J. A. García-Espín, "Virtual network function scheduling: Concept and challenges," in *2014 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, June 2014, pp. 1–5.

[4] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

[5] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 1346–1354.

[6] 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.

[7] 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.

[8] J. Guo, F. Liu, J. C. S. Lui, and H. Jin, "Fair network bandwidth allocation in iaas datacenters via a cooperative game approach," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 873–886, 2016.

[9] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746–3758, 2016.

[10] 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, Sep. 2017.

[11] D. Liao, Y. Wu, Z. Wu, Z. Zhu, W. Zhang, G. Sun, and V. Chang, "Ai-based software-defined virtual network function scheduling with delay optimization," *Cluster Computing*, vol. 22, no. 6, pp. 13 897–13 909, 2019.

[12] D. Zhao, J. Ren, R. Lin, S. Xu, and V. Chang, "On orchestrating service function chains in 5g mobile network," *IEEE Access*, vol. 7, pp. 39 402–39 416, 2019.

[13] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: ACM, 1967, pp. 483–485. [Online]. Available: http://doi.acm.org/10.1145/1465482.1465560

[14] M. D. McCool, A. D. Robison, and J. Reinders, *Structured parallel programming: patterns for efficient computation*, 1st ed. Elsevier, 2012.