Location-based Data Model for Optimized Network Slice Placement

José Jurandir Alves Esteves, Amina Boubendir, Fabice Guillemin, Pierre Sens

Location-based Data Model for Optimized Network Slice Placement

José Jurandir Alves Esteves*†, Amina Boubendir*, Fabrice Guillemin* and Pierre Sens†
*Network Architecture & Automation department, Orange Labs, France
†CNRS, INRIA, LIP6, Sorbonne Université, France
{josejurandir.alvesesteves, amina.boubendir, fabrice.guillemin}@orange.com, pierre.sens@lip6.fr

***Abstract*—Network Slicing has its roots in Network Function Virtualization (NFV) allowing high flexibility in the delivery of end-to-end network services. To achieve Network Slicing promises on efficiency, Network Slice Providers have to ensure optimized resource utilization and to guarantee Quality of Service when managing the life-cycle of a Network Slice. We focus in this paper on Network Slice Placement, intimately related to the VNF Placement and Chaining problem. In contrary to most studies related to VNF placement, we deal with the most complete and complex Network Slice topologies and we pay special attention to the geographic location of Network Slice Users. We propose a data model adapted to Integer Linear Programming. Extensive numerical experiments assess the relevance of taking into account the user location constraints.**

***Index Terms*—NFV, Network Slicing, Optimization, Data Models, Placement Algorithms, Service Functions Chains.**

I. Introduction

Network Functions Virtualization (NFV) introduces high flexibility in the implementation of Network Functions (NFs) by breaking the coupling between NF logic and the hosting hardware so that Virtualized Network Functions (VNFs) can be deployed on commercial off-the-shelf servers [1]. As a corollary, NFV enables the management of the life-cycle of VNFs independently from the underlying physical infrastructure. But, first and foremost, NFV enables deploying multiple logical networks (as a series of VNFs) over the same Physical Substrate Network (PSN) shared between the latter.

This gives rise to the concept of Network Slicing. It is known that Network Slicing takes benefit of the logical and/or physical separation of network resources to allow multi-tenancy support, customization and isolation of Network Slices [2]. However, there are various definitions of Network Slicing available in the literature.

The NGMN proposed a definition of a Network Slice Instance (NSI) as a set of Network Slice Subnet Instances (NSSI), each comprising a set of NFs that are needed to implement the desired Network Slice functionalities [3]. 3GPP adds the idea of completeness of an NSI. This means that an NSI must include all NFs and resources necessary to support the set of communications it offers [4]. ETSI proposes a mapping between the 3GPP Network Slice concept and the Network Service concept [1], [5]. From the ETSI point of view, a Network Service is a resource-centric view of an NSI. We use in the following the ETSI Network Slice view.

Proper management and orchestration of Network Slices, VNFs and their associated Virtual Links (VLs) are essential to achieve Network Slicing. However, guarantee optimized resource and Quality of Service (QoS) when managing the life-cycle of a VNF or a Network Slice remains a challenge.

VNF Placement and Chaining (VNF-PC) problem is in fact an optimization problem falling into the broad family of resource allocation problems. It consists of choosing the servers of the PSN in which the VNFs composing Network Slices are to be deployed and which physical links to use in order to steer traffic between this servers. This problem contains a specific optimization objective (e.g., minimizing resource consumption, optimizing a specific QoS metric, etc.) that must be satisfied [6] [7].

In spite of the various papers about VNF-PC and its variants, most of them frequently ignores the VNF-PC geographic dimension. Existing studies often do not take into account neither the user's location when solving VNF-PC nor user location implications, especially in the end-to-end (E2E) delay calculation. We specifically address this challenge here.

In this context, the contributions of this paper are as follows:

1) Propose an E2E delay model that integrates the user location as an end-point of the Network Slice, and improves scalability by exploiting the possibility of grouping Network Slice Users (NSU) instead of considering them individually;
2) Deal with the most complex Network Slice topologies, going beyond the currently studied Service Functions Chain (SFC) concept;
3) Set no restrictions on the placement location of two VNFs of the same Network Slice.

The proposed model is formalized mathematically using Integer Linear Programming (ILP). We use CPLEX solver to assess the performance of the model.

This paper is organized as follows. In Section II we analyze the related work. Section III provides all the assumptions and definitions. Section IV introduces the proposed ILP used to solve the Network Slice Placement problem. The numerical experiments and evaluation results are presented in Section V. Some concluding remarks and perspectives are given in Section VI.

II. Related Work Analysis

We analyze in this Section recent works on Network Slice Placement and more generally Service Functions Chains placement and VNF Forwarding Graph/Virtual Network Embedding according to four main aspects as detailed hereafter.

A. *VNF Forwarding Graph Embedding*

With the emergence of the Network Slicing, the Network Slice Placement problem has become critical for network operators. This problem is intimately related to VNF Forwarding Graph Embedding (VNF-FGE), Virtual Network Embedding (VNE) and Service Function Chain placement (SFC-P) problems, which have been treated in the framework of VNF-PC.

VNF-FGE is seen as a generalization of the SFC-P and VNE since it tackles more complex placement requests in terms of topology and considers allocation of multiple resources simultaneously [7]. VNE problem is $\mathcal{NP}$-hard [8]. Since VNF-FGE is a generalization of VNE, then, VNF-FGE is also $\mathcal{NP}$-hard. We consider numerous papers dealing with SFC-P, VNE, and VNF-FGE problems [7], [9]–[11]. Different heuristics have been proposed to solve these problems, mainly by dealing with their online versions [6], [12]–[14].

Heuristic algorithms are frequently used for reducing resolution time in large scale scenarios. We have instead used ILP since it allows us to unambiguously define our data model and to obtain proven optimal solutions in small scale scenarios.

B. *Optimization targets and constraints*

Existing models cope with different optimization targets and constraints. However, most authors have focused on resource consumption rather than on QoS.

Reference [6] propose an online Eigen decomposition approach to solve VNF-FGE. They show the scalability of their approach via simulations taking into account Physical Substrate Networks with up to 5000 nodes. QoS constraints (e.g. latency, delay) are not considered in their work.

In [12], the authors propose a ”boosted ILP” to solve VNF-FGE. They allow VNFs to be shared between different VNF-FGs and their objective is to minimize power consumption. However, QoS constraints are not explicitly treated. In [13], an ILP strategy is used to solve VNF-FGE allowing VNF replications. The authors optimize load balancing and resources utilization but do not deal with any QoS parameter.

In [14], distributed optimization strategies are used to solve VNF-PC. They treat the placement of E2E services in a Physical Substrate Network composed of multiple administrative domains but without including important QoS criteria like E2E delay.

The present work focuses on a specific QoS parameter that is E2E delay due to its critical importance in Network Slicing (e.g. ultra-low latency communications in 5G use-cases [15]).

C. *Delay-aware placement*

Researchers have used different approaches to deal with E2E delay while studying VNF-PC. In [16] average E2E delay is studied as an optimization criteria on SFC-P.

However, meeting E2E delay requirements is not ensured when service E2E delay is treated as an optimization target rather than a strict constraint.

The authors of [17] deal with E2E delay as a strict constraint and they propose a Mixed Integer Quadratically Constrained Program for SFC placement. However, as in most of the cases, they do not consider user location as an end-point of the Network Slice. This is utmost important in Network Slice Placement but has received so far less attention by the community.

D. *Location-aware placement*

The authors of [18] were the first ones to propose an approach to solving VNE taking into account user location. Their algorithms are based on the assumption that a preferred placement location is known for each Virtual Node of the Virtual Networks to be placed. This assumption is reused in [19], where the authors propose the transformation of the location-constrained VNE into a minimum cost maximum clique problem. Their work is original but the assumption that two virtual nodes of the same Virtual Network cannot be embedded on the same machine is structuring for the proposed algorithms. Hence their applicability is limited in the context of Network Slicing.

The assumption of [18] was also recently further exploited in VNF-FGE. In [20] the authors propose algorithms for solving the Service Embedding and Chain Composition problem. This problem is defined by jointly solving the VNF-FG Composition (VNF-FGC) and the VNF-FGE. The VNF-FGC problem is another optimization problem in the family of NFV resource allocation problems. It is about defining a suitable VNF-FG for offering a specific service by taking as input a set of VNFs and order constraints for them [7]. In spite of the originality of their work, the authors contemplate trees as the only possible topologies for VNF-FG. Hence, the proposed algorithms cannot deal with cyclic VNF-FGs, which limits its applicability to Network Slicing.

The preferred location assumption regarded in the above three works have the advantage of reducing the problem complexity. However, the accurate determination of the preferred placement location for each Virtual Node while solving VNE or VNF-FGE is not straightforward. To address this issue, we propose an E2E delay model that look at the user location as an end-point of the Network Slice. In addition, to be more generic, we do not make any assumption on the embedding location of two nodes of the same Network Slice (unlike [19]).

The closest related work to the study developed in this paper is [21]. The authors propose an ILP and a heuristic to solve a joint user association and SFC placement problem. Their work is original, but our approach tackles more complex and complete Network Slicing use cases than SFC. Also, by considering each UE that would be connected to a SFC the authors increase drastically the number of binary variables of their model in comparison to ours. We aim at overcoming these scalability issues by exploring the possibility of grouping Network Slices Users.

TABLE I: High level data model elements description

Model domains	**Model elements**	**Model sub-elements**
Physical Substrate Network (PSN)	Virtualized Infrastructure (VI)	Data centers (DC)
		Server (SE) and switch (SW)
		Data center link (DL)
	Transport Network (TN)	Router (R)
		Transport link (TL)
	Access Network (AN)	User access point (UAP)
		Access link (AL)
Network Slice (NS)	Network Slice Provider (NSP)	Network Slice User (NSU)
	Network Slice Placement Request (NSPR)	VNFs
		Virtual link (VL)
		E2E chain (E2EC)

III. Definitions & Assumptions

In this section, we define the key components of the proposed model to solve the Network Slice Placement problem. The elements and their acronyms are summarized in Table I.

A. The Virtualized Infrastructure

The Virtualized Infrastructure represents the set of data centers that provide the physical and virtual resources in the form of various servers containing computing and storage resources to support a Network Slice deployment. These resources are virtualized by means of hypervisors allowing the execution of multiple virtualized components (e.g., VMs, OS containers). Switches ensures the connectivity between these servers and with the Transport Network.

B. The Transport Network and Access Network

The Virtualized Infrastructure is distributed in different Points of Presence (PoP) and connected by a Transport Network composed of a set of routers and transport links. The users of each Network Slice access the corresponding services via a UAP (e.g., Wi-fi access point, cellular antenna). We consider the cases where the NSUs for a given Network Slice are located nearby the respective UAP and that their location does not change in time (e.g., events on stadiums or airports, industry 4.0 use cases). This does not prescribe an individual user to move from one UAP to another.

C. The Physical Substrate Network model

The PSN is modeled as an undirected graph as illustrated in Figure 1. In such a graph, the nodes represent the UAPs, routers, switches and servers. The edges represent the links between them. Each node is labeled with a type in {UAP, router, switch, server}.

The nodes of type server are labeled with a CPU and RAM capacity. Transport links and data center links are labeled with a bandwidth capacity and an induced delay. Access links are labeled with an induced delay. The delay induced between each UAP and each server is called Access Delay. This metric is illustrated in Figure 1 for servers S2, S3 and S6.

Some assumptions are made for our PSN model. First, we assume that the cost of a server is the same for any server, and the cost of a link is the same for any link. This may not be the case if we consider that the different data centers available are not owned by the Network Slice Provider.

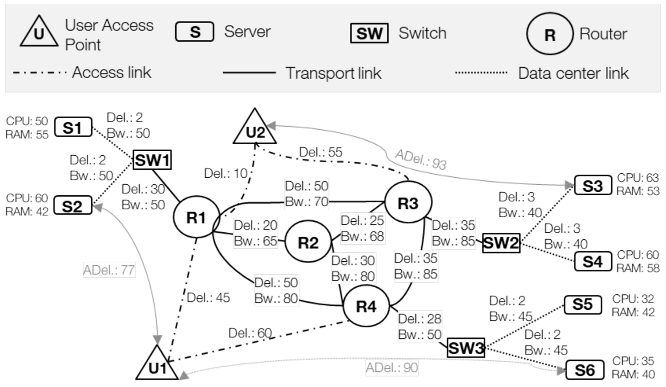


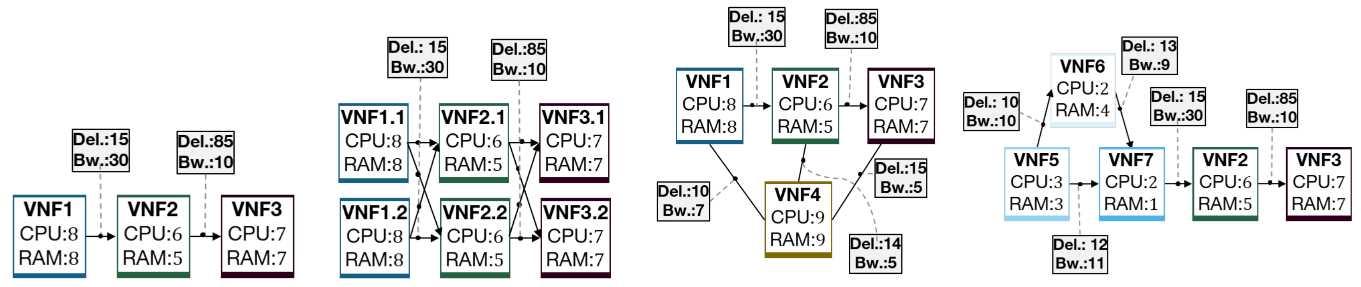
Fig. 1: Example of Physical Substrate Network modeled as an undirected graph.

Also, we assume that all the physical links are bidirectional and offer the same capacities when used for uplink and downlink. As the communication delay between two VNFs placed in the same server is not of the same order of magnitude as when they are placed in different servers, we assume that it can be neglected. For the sake of simplicity, we consider that the delays induced by physical links are constant. Finally, we assume that the number of VNFs that can be deployed inside the same server is only bounded by the amount of CPU and RAM available in this machine.

D. The Network Slice

A Network Slice is offered to a specific group of NSUs by an entity called Network Slice Provider. A NSPR represents a view of the resources requirements for a Network Slice to be placed in the PSN. We model an NSPR as an undirected graph in which the nodes represent the VNFs composing the Network Slice and the edges the VLs between them. The nodes of the graph are labeled with a CPU and RAM requirement and the edges are labeled with a bandwidth and communication delay requirement (see Figures 2 and 3).

Most of the works on VNF-PC adopt the concept of SFC to represent what would be a Network Slice. However, SFC is appropriate only for the case where the Network Slice is described by an ordered sequence of user plane VNFs (e.g., [21]).



(a) Service Functions Chain.

(b) Network Slice with VNF and VL redundancy.

(c) Network Slice with a centralized control plane VNF.

(d) Network Slice with a VNF implemented using a non-monolithic architecture.

Fig. 2: Network Slice vs Service Functions Chain.

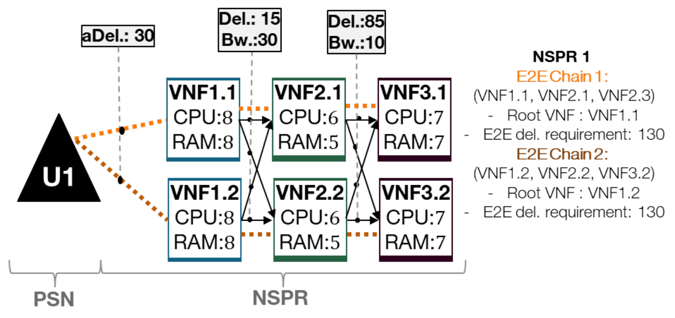


Fig. 3: Example of Network Slice Placement request and corresponding User Access Point

However, it is not enough to model at least 3 characteristics a Network Slice may have, as illustrated in Figure 2.

The first one is high availability that is ensured via redundancy of VNFs and VLs (Figure 2(b)). Another characteristic is separation between user plane and control plane VNFs. A simple example of Network Slice comprising a centralized control plane function (VNF4) that is connected to the user plane functions is captured in Figure 2(c).

The third one is when at least one VNF of the Network Slice is implemented using a non-monolithic architecture (e.g micro-services architecture). An example is given on Figure 2(d), where VNF1 of Figure 2(a)) is decomposed into VNFs 5, 6 and 7. These examples illustrate the importance of considering Network Slice as a generalization of SFC.

Another particularity of the proposed NSPR model is the concept of E2E Chain. As presented in Figure 3, the E2E Chains (E2EC) of NSPRs are defined by the sequences of VNFs traffic can traverse. We name the first VNF of each E2EC by root VNF.

Each group of NSUs imposes a maximum acceptable Access Delay between the UAP users are connected to and the root VNFs of the NSPR they request in order to ensure the feasibility of the communication. The E2E delay requirement for each E2EC of one NSPR stands for the maximum delay allowed between the UAP associated to the NSPR and the last VNF of the E2EC.

E. Network Slice Placement Problem

We summarize the Network Slice Placement problem as follows:

- *Given:* a set of NSPRs to be offered by a NSP; a PSN,
- *Find:* in which server of the PSN to instantiate each requested VNF; which physical links to use in order to realize the VLs requested between these VNFs,
- *Subject to:* servers CPU and RAM available capacity, physical links bandwidth available capacity, link delay, access delay, and E2E delay requirements of each NSPR,
- *Objective:* minimizing total resource consumption.

IV. Mathematical Problem Modeling

In this section, we present the ILP formulation to solve the Network Slice Placement problem described in Section III. The complete data model is illustrated in Figure 4.

A. Notations

- **Physical Substrate Network**
  - $N \subset \mathbb{N}^{|\mathbb{N}|}$: set composed by the routers, switches and servers
  - $S \subset N$: set of servers
  - $L = \{(a,b)|(a,b) \in N^2 \wedge a \neq b\}$: set of physical links
  - $cap^{bw}_{(a,b)} \in \mathbb{R}, \forall (a,b) \in L$: bandwidth capacity of physical link (a,b)
  - $cap^{cpu}_s \in \mathbb{R}, \forall s \in S$: CPU capacity of server s
  - $cap^{ram}_s \in \mathbb{R}, \forall s \in S$: RAM capacity of server s
  - $\delta_{(a,b)} \in \mathbb{R}, \forall (a,b) \in L$: delay induced by physical link (a,b)

- **Network Slice Placement requests**
  - $R \subset \mathbb{N}^{|R|}$: set of NSPRs to place
  - $N^r \subset \mathbb{N}^{|N^r|}, \forall r \in R$: set of VNFs of the NSPR r
  - $E^r = \{(\bar{a},\bar{b})|(\bar{a},\bar{b}) \in N^r \times N^r \wedge \bar{a} \neq \bar{b}\}, \forall r \in R$: set of virtual links of the NSPR r
  - C^r $\forall r \in R$: set of E2E chains on the NSPR r[1]
  - $n^{r,c}_{root} \in N^r, \forall r \in R, \forall c \in C^r$: the root VNF of E2E chain c of NSPR r

[1]Each E2E chain is described by an ordered list of virtual links.

- $d_n^{cpu} \in \mathbb{R}, \forall r \in R, \forall n \in N^r$: CPU requirement of VNF n
- $d_n^{ram} \in \mathbb{R}, \forall r \in R, \forall n \in N^r$: RAM requirement of VNF n
- $d_{(\bar{a},\bar{b})}^{bw} \in \mathbb{R}, \forall r \in R, \forall (\bar{a},\bar{b}) \in E^r$: bandwidth requirement of VL $(\bar{a},\bar{b})$
- $d_{(\bar{a},\bar{b})}^{\delta} \in \mathbb{R}, \forall r \in R, \forall (\bar{a},\bar{b}) \in E^r$: delay requirement of VL $(\bar{a},\bar{b})$
- $\delta_c^r \in \mathbb{R}, \forall r \in R, \forall c \in C$: maximum E2E delay accepted for E2E chain c of NSPR r
- $\alpha_{max}^r \in \mathbb{R}, \forall r \in R$: maximum access delay accepted for the E2E chains of NSPR r
- $\alpha_s^r \in \mathbb{R}, \forall r \in R, \forall s \in S$: access delay between the user access point of NSPR r and server s

B. Decision variables

- $x_s^n \in \{0,1\}, \forall r \in R, \forall n \in N^r, \forall s \in S$: equals 1 if the VNF n is placed into server s and 0 otherwise
- $y_{(a,b)}^{(\bar{a},\bar{b})} \in \{0,1\}, \forall r \in R, \forall (\bar{a},\bar{b}) \in E^r, \forall (a,b) \in L$: equals 1 if the virtual link $(\bar{a},\bar{b})$ is mapped into physical link (a,b) and 0 otherwise

C. Integer Linear Program

The integer linear program formulation is as follows.

$$\min_{x,y} \sum_{r\in R}\sum_{n\in N^r}\sum_{s\in S} \frac{d_n^{ram} x_s^n}{cap_s^{ram}} + \sum_{r\in R}\sum_{n\in N^r}\sum_{s\in S} \frac{d_n^{cpu} x_s^n}{cap_s^{cpu}} + \sum_{r\in R}\sum_{(\bar{a},\bar{b})\in E^r}\sum_{(a,b)\in L} \frac{d_{(\bar{a},\bar{b})}^{bw} y_{(a,b)}^{(\bar{a},\bar{b})}}{cap_{(a,b)}^{bw}} \quad (1)$$

subject to:

$$\sum_{s\in S} x_s^n = 1 \qquad \forall r \in R, \ \forall n \in N^r \quad (2)$$

$$\sum_{r\in R}\sum_{n\in N^r} d_n^{cpu} x_s^n \le cap_s^{cpu} \qquad \forall s \in S \quad (3)$$

$$\sum_{r\in R}\sum_{n\in N^r} d_n^{ram} x_s^n \le cap_s^{ram} \qquad \forall s \in S \quad (4)$$

$$\sum_{r\in R}\sum_{(\bar{a},\bar{b})\in E^r} d_{(\bar{a},\bar{b})}^{bw} y_{(a,b)}^{(\bar{a},\bar{b})} + d_{(\bar{a},\bar{b})}^{bw} y_{(b,a)}^{(\bar{a},\bar{b})}) \le cap_{(a,b)}^{bw} \qquad \forall (a,b) \in L \quad (5)$$

$$y_{(a,b)}^{(\bar{a},\bar{b})} + y_{(b,a)}^{(\bar{a},\bar{b})} \le 1 \qquad \forall r \in R, \ \forall (\bar{a},\bar{b}) \in E^r, \ \forall (a,b) \in L \quad (6)$$

$$\sum_{\substack{b\in N:\\(a,b)\in L}} y_{(a,b)}^{(\bar{a},\bar{b})} - \sum_{\substack{b\in N:\\(b,a)\in L}} y_{(b,a)}^{(\bar{a},\bar{b})} = x_a^{\bar{b}} - x_a^{\bar{a}} \qquad \forall a \in S, \ \forall r \in R, \ \forall (\bar{a},\bar{b}) \in E^r \quad (7)$$

$$\sum_{\substack{b\in N:\\(a,b)\in L}} y_{(a,b)}^{(\bar{a},\bar{b})} - \sum_{\substack{b\in N\\(b,a)\in L}} y_{(b,a)}^{(\bar{a},\bar{b})} = 0 \qquad \forall a \in N \setminus S, \ \forall r \in R, \ \forall (\bar{a},\bar{b}) \in E^r \quad (8)$$

$$\sum_{(a,b)\in L} \delta_{(a,b)} y_{(a,b)}^{(\bar{a},\bar{b})} \le d_{(\bar{a},\bar{b})}^{\delta} \qquad \forall r \in R, \ \forall (\bar{a},\bar{b}) \in E^r \quad (9)$$

$$\sum_{s\in S} \alpha_s^r x_s^{n_c} \le \alpha_{max}^r \qquad \forall r \in R, \ \forall c \in C^r \quad (10)$$

$$\sum_{s\in S} \alpha_s^r x_s^{n_c} + \sum_{(a,b)\in L}\sum_{(\bar{a},\bar{b})\in c} \delta_{(a,b)} y_{(a,b)}^{(\bar{a},\bar{b})} \le \delta_c^r \qquad \forall r \in R, \ \forall c \in C^r \quad (11)$$

The objective function given by Equation (1) is to minimize the global resources consumption weighted by the their scarcity in order to differentiate the cost of a resource from another. In this way, the the scarcest resources are considered more expensive than the most abundant.

The function is composed of three parts: the sum of the RAM, CPU and bandwidth consumption, respectively. Constraints (2) ensures the placement of each VNF of each NSPR in only one server. Constraints (3) and (4) ensure that the resource (CPU and RAM, respectively) capacity of each server is not exceeded by the resource requirements of the VNFs placed in the server.

Constraints (5) guarantee that the bandwidth capacity of each physical link is not exceeded by the requirements of the virtual links mapped. Constraints (6) ensure that each physical link used by a VL is used in only one direction. Multi-commodity flow constraints (7) and (8) ensures that if two connected VNFs are mapped in different servers, the VL that connects them is mapped into one physical path between these two servers.

Constraints (9) guarantee that the delay requirements of each virtual link of each NSPR is respected. Finally, constraints (10) make sure that the access delay is respected for each NSPR and constraints (11) ensure the fulfillment of the required E2E delay for each E2E chain in each NSPR.

V. Experiments & Evaluation Results

We have implemented the proposed ILP formulation in Julia [22] and used the default branch-and-bound algorithm from `ILOG` CPLEX 12.9 solver [23]. We used a 2x6 cores @2.95Ghz CPU machine with 96GB of memory in our experiments. To allow extensive simulations we stop CPLEX execution when the gap is lower than 1% or after two hours.

A. Experimentation Settings

We designed a random parameter generator based on the GT-ITM tool [24] to generate different simulation scenarios.

TABLE II: VI input elements to model parameters generator and parameters generated to model execution

Input elements to model parameters generator	Input values to model parameters generator	Parameters generated to model execution
# of servers in each DC # of switches in each DC	5 1	List of identifiers of servers in each DC List of identifiers of switches in each DC
Grid for servers and switches coordinates Min. and max. CPU capacity for servers Min. and max. RAM capacity for servers	100×100 [50,100] [50,100]	Servers locations (cartesian coordinates) Switches locations (cartesian coordinates) Servers CPU capacity Servers RAM capacity
Link existence probability Min. and max. bandwith capacity	1 [50,500]	List of identifiers of links between servers and switches Links bandwidth capacity Delay induced by each link (euclidian distance)

TABLE III: TN and AN input elements to model parameters generator and parameters generated to model execution

Input elements to model parameters generator	Input values to model parameters generator	Parameters generated to model execution
# of routers Grid for routers coordinates	5, 10, 20 and 40 100×100	List of identifiers of routers Routers locations (cartesian coordinates)
Link existence probability Min. and max. bandwidth capacity	1 [50,500]	List of identifiers of links between routers Links bandwidth capacity Delay induced by each link (euclidian distance)
# of DCs to which each router is connected Min. and max. bandwidth capacity	1 [50,500]	List of identifiers of links between routers and DCs Links bandwidth capacity Delay induced by each link (euclidian distance)
Grid for UAP coordinates	100×100	UAP location (cartesian coordinates) Delay between UAP and each server (euclidian distance)

TABLE IV: Network Slice input elements to model parameters generator and parameters generated to model execution

Input elements to model parameters generator	Input values to model parameters generator	Parameters generated to model execution
# of VNFs Grid for VNFs pseudo coordinates Min. and max. CPU requirement Min. and max. RAM requirement	3, 5, 10, 15 and 20 100×100 * *	List of identifiers of VNFs VNFs pseudo coordinates VNFs CPU requirement VNFs RAM requirement
VLs existence probability Min. and max. bandwidth requirement	0.5 [5,10]	List of identifiers of VLs between VNFs VLs bandwidth requirement Delay required by each VL (euclidian distance)
Min. # of VNFs by E2EC Max. # of end E2EC by NSPR	3 10	List of identifiers of E2EC List of identifiers of VNFs in E2EC Root VNF of E2EC Delay required between UAP and root VNF E2E delay required by each E2EC

Tables II-IV summarize the generator inputs and outputs. Each simulation scenario is defined by a PSN and a set of NSPRs. A PSN is generated as a Transit-stub graph [25]. Transport Network is represented by a transit domain with a certain number of routers. Each datacenter is represented by a stubdomain composed by 5 servers connected to a switch. Each switch is connected to a Transport Network router. Resource capacities of servers and physical links are generated randomly in the intervals specified in Tables II and III.

A grid is used to generate the locations of physical nodes as Cartesian coordinates. The delay induced by the physical links is given by the Euclidian distance between the nodes. An NSPR is generated as a random graph. In each simulation scenario, we consider 5, 10, 20, 40, 80, or 100 NSPRs to be placed each one having a specific specific requirements for VNFs and VLs. The resource requirements of VNFs and VLs are generated randomly within an interval.

This interval is fixed for bandwidth requirement generation (see Table IV) but varies according to the scenario for CPU and RAM requirements generation.

The lower bounds (Lb) and upper bounds (Ub) of these intervals are defined by Equation (12), where $nVNFs$ represent the number of VNFs per NSPR in the scenario, S the set of servers and cap_s^j refers to the capacity of resource $j \in \{cpu, ram\}$ for server $s \in S$.

$$\begin{aligned} Ub_j &= \min_{s \in S} cap_s^j \\ Lb_j &= \begin{cases} \frac{\max_{s \in S} cap_s^j}{nVNFs} + 1, & \text{if } \frac{\max_{s \in S} cap_s^j}{nVNFs} + 1 \leq Ub_j \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

To compute VL delay requirements, we generate Cartesian coordinates for the VNFs of each NSPR and calculate the Euclidean distance between them.

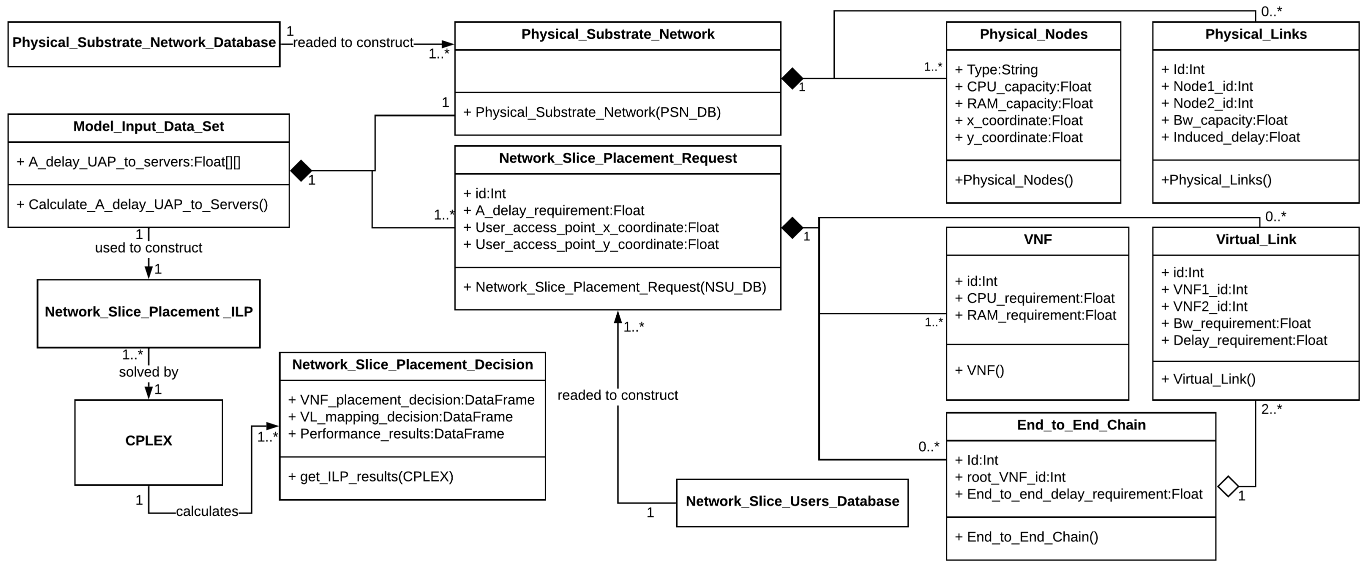

Physical_Substrate_Network_Database
readed to construct
Physical_Substrate_Network
+ Physical_Substrate_Network(PSN_DB)
Physical_Nodes
+ Type:String
+ CPU_capacity:Float
+ RAM_capacity:Float
+ x_coordinate:Float
+ y_coordinate:Float
+Physical_Nodes()
Physical_Links
+ Id:Int
+ Node1_id:Int
+ Node2_id:Int
+ Bw_capacity:Float
+ Induced_delay:Float
+Physical_Links()
Model_Input_Data_Set
+ A_delay_UAP_to_servers:Float[][]
+ Calculate_A_delay_UAP_to_Servers()
used to construct
Network_Slice_Placement_Request
+ id:Int
+ A_delay_requirement:Float
+ User_access_point_x_coordinate:Float
+ User_access_point_y_coordinate:Float
+ Network_Slice_Placement_Request(NSU_DB)
VNF
+ id:Int
+ CPU_requirement:Float
+ RAM_requirement:Float
+ VNF()
Virtual_Link
+ id:Int
+ VNF1_id:Int
+ VNF2_id:Int
+ Bw_requirement:Float
+ Delay_requirement:Float
+ Virtual_Link()
Network_Slice_Placement _ILP
solved by
CPLEX
calculates
Network_Slice_Placement_Decision
+ VNF_placement_decision:DataFrame
+ VL_mapping_decision:DataFrame
+ Performance_results:DataFrame
+ get_ILP_results(CPLEX)
readed to construct
Network_Slice_Users_Database
End_to_End_Chain
+ Id:Int
+ root_VNF_id:Int
+ End_to_end_delay_requirement:Float
+ End_to_End_Chain()
1
1..*
0..*
2..*

Fig. 4: Data model representation using UML class diagram

The E2E delay requirement for each E2EC is the sum of the delay requirements of the VLs composing the chain. The Access Delay requirement for a given NSPR is the percentile 60 of an ordered vector containing the delays between UAP associated to the the NSPR and the servers. A set of 10 random scenarios was generated for each combination of the input values to model parameters generator detailed in Tables II-IV.

B. *Evaluation Metrics*

We examine two types of performance metric.

1) Scalability: To evaluate the scalability of our approach we adopt two metrics: the model execution time in seconds and the estimated gap to optimal in % after two hours.

2) Relevance of the E2E delay model: We compare our location-based model with a location-agnostic model widely used in the state-of-the-art, that considers E2E delay but does not take user location into account (e. g., [17]). This comparison is done using two metrics: E2E delay requirement violation and average E2E delay requirement violation. These metrics are defined by Equations (13) and (15).

Let Δ_c^r and $\bar{\Delta}_c^r$ $\forall r \in R$, $\forall c \in C^r$ be the E2E delays calculated by the location-based and location-agnostic models, respectively. E2E delay requirement violation for each E2EC is given by Eq. (13). E2E delay requirement violation metric for each simulation scenario m is given by Eq. (14). The average E2E delay requirement violation metric is given by Eq. (15).

$$\epsilon_c^r = \begin{cases} \max(0, \frac{\Delta_c^r - \delta_c^r}{\delta_c^r}), & \text{if location-based model} \\ \max(0, \frac{(\bar{\Delta}_c^r + \alpha_{max}^r) - \delta_c^r}{\delta_c^r}, & \text{otherwise} \end{cases} \tag{13}$$

$$\epsilon_m = \sum_{r \in R} \frac{1}{|C^r|} \sum_{c \in C^r} \epsilon_c^r \tag{14}$$

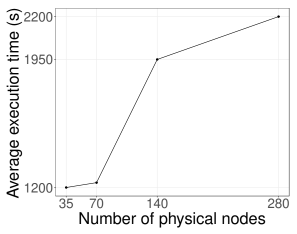
$$\bar{\epsilon} = \frac{1}{M} \sum_{m=1,\dots,M} \epsilon_m \tag{15}$$

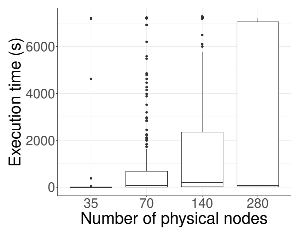
C. *Evaluation Results & Discussions*

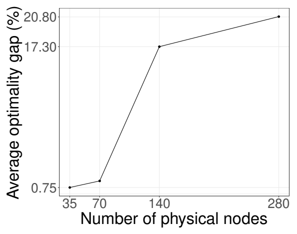
Figures 5(a) and 5(c) respectively show the evolution of the average execution time of our model and the average gap to optimal estimated by CPLEX at the end of the execution time in function of the number of Physical Substrate Network's nodes. These two metrics significantly increase for the scenarios with a Physical Substrate Network with more than 70 nodes due to complexity explosion. The variation of execution times and optimality gaps for all solved simulation scenarios are illustrated in Figures 5(b) and 5(d), respectively. We note high variation in the execution times associated with the different levels of complexity of simulation scenarios. Optimality gaps are often at 0% showing that the algorithm converged in 2 hours in most simulations.

Figure 7 shows the average resource consumption of the proposed solutions. We remark that the RAM and CPU consumption are always equal to 100% of the required since the mapping of all VNFs is mandatory. However, we notice an average bandwidth economy of at least 20% of the required due to the placement of multiple VNFs of the same NSPR inside a same server. The pertinence of the proposed E2E delay model is analyzed in Figures 6(a)-(d). We compare E2E delay requirement violations obtained when we solve our ILP (User access point location is considered) and when we solve the alternative ILP (User access point location is not considered).

Figure 6(a) presents the evolution of the average E2E delay requirement violation according to the number of nodes on the PSN. In contrast to our formulation that always respects E2E delay requirements, a growing average E2E delay requirement violation is observed when we do not take into account user location.


(a) Average execution time.


(b) Execution time variation.


(c) Average optimality gap.

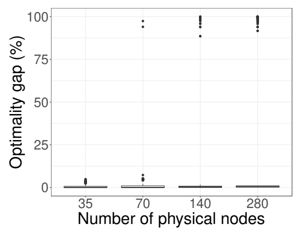
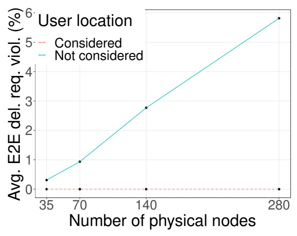
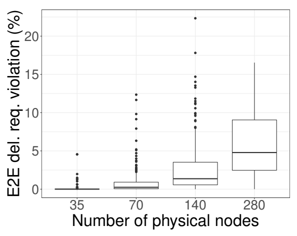
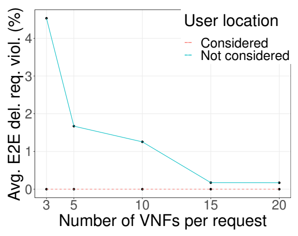

(d) Optimality gap variation.

Fig. 5: Scalability evaluation results.


(a) Average E2E delay requirement violation.


(b) E2E delay requirement violation variation.


(c) Average E2E delay requirement violation.

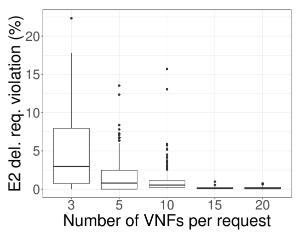

(d) E2E delay requirement violation variation.

Fig. 6: E2E delay model relevance analysis.

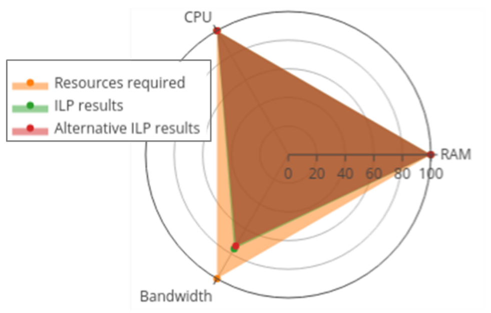


Fig. 7: Resource consumption evaluation: CPU, RAM, BW

We observe a high variation of the E2E delay requirement violations captured by Figure 6(b) as we notice that it may exceed 15% in some cases. Figure 6(c) presents the evolution of the average of the E2E delay requirement violations according to the number of VNFs of the NSPRs.

We also observe that the average violation decreases while the number of VNFs in the NSPRs increases. This happens because the more VNFs we have in the NSPRs, less the NSPRs are spread in the PSN, this reduces the E2E delay requirements violations. In fact, the CPU and RAM requirements of each VNF decrease when the number of VNFs per request increases (see Equation (12)). Hence, the tested models concentrate more VNFs inside the same machines to prevent from using link resources. Observing the variation of the E2E delay requirement violations in Figure 6(d), we see again a high deviation of the violation results still higher than the median up to 20%.

VI. Conclusions & Future Work

We have studied the Network Slice Placement problem focusing on the constraints implied by user location. The proposed ILP solution gathers four main contributions. First, a new E2E delay model integrating the user location as an end-point of the Network Slice without assuming that the preferred location for each Virtual Node is known (generalizing the hypothesis introduced by [18]). Second, a model that can considers a higher number of Network Slice Users and explore the possibility of grouping them (unlike [21]) for scalability issues. Moreover, modeling Network Slices as a generalization of SFCs since they can have a cyclic topology (generalizing at least [20] and [21] hypotheses). Finally, removing all restrictions on the placement location of two VNFs of the same Network Slice (generalizing [19] hypothesis).

We have implemented the proposed ILP formulation and used the default branch-and-bound algorithm from `ILOG` CPLEX 12.9 solver to do extensive simulations in order to evaluate our proposal. The results show the relevance of the proposed E2E delay model since the simulations showed that taking into account the user location as an end point of the network slice is essential in order to ensure the fulfillment of the E2E delay requirements.

By solving the different simulation scenarios, we were able to see that we can reach 5% of average E2E delay requirement violation and up to 20% when we do not include the user location to calculate the Network Slice placement decision.

As perspectives for future work, we plan to develop heuristic algorithms to overcome complexity explosion when solving large scale scenarios and implement an online scheme based in this formulation in order to investigate dynamic aspects of VNF Placement and Chaining.

Acknowledgment

The presented experiments were carried out using Grid'5000 testbed supported by a scientific interest group at Inria and Sorbonne University (www.grid5000.fr).

References

[1] ETSI NFV ISG, "Network Functions Virtualization (NFV); Architectural Framework, ETSI Standard GS NFV 002 V1.2.1," 2014. [Online]. Available: https://www.etsi.org/technologies-clusters/technologies/nfv.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surv. Tut.*, vol. 20, no. 3, pp. 2429–2453, 2018.

[3] NGMN Alliance, "Description of network slicing concept," *NGMN 5G P*, 2016.

[4] 3GPP, "Telecommunication management; study on management and orchestration of network slicing for next generation network," TR 28.801 V15.0.0, 2017, Available at: https://www.3gpp.org/specifications, Tech. Rep., 2017.

[5] ETSI NFV ISG, "Network Functions Virtualisation (NFV); Evolution and Ecosystem; Report on Network Slicing Support, ETSI Standard GR NFV-EVE 012 V3.1.1," 2017. [Online]. Available: https://www.etsi.org/technologies-clusters/technologies/nfv.

[6] M. Mechtri, C. Ghribi, and D. Zeghlache, "Vnf placement and chaining in distributed cloud," in *Proc. 9th IEEE Int. Conf. Cloud Comput.*, 2016, pp. 376–383.

[7] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, 2016.

[8] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electron. Notes in Discrete Mathematics*, vol. 52, pp. 213–220, 2016.

[9] A. Fischer, J. F. Botero Vega, M. Duelli, D. Schlosser, X. Hesselbach Serra, and H. De Meer, "Alevin-a framework to develop, compare, and analyze virtual network embedding algorithms," *Open Access J. Electron. Commun. EASST*, pp. 1–12, 2011.

[10] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surv. Tut.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[11] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surv. Tut.*, vol. 21, no. 2, pp. 1409–1434, 2018.

[12] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "A green vnf-fg embedding algorithm," in *Proc. 4th IEEE Conf. Netw. Softwarization*, 2018, pp. 141–149.

[13] F. Carpio, W. Bziuk, and A. Jukan, "Replication of virtual network functions: Optimizing link utilization and resource costs," in *Proc. 40th IEEE Int. Convent. Inf. Commun. Technol., Electronics and Microelectronics*, 2017, pp. 521–526.

[14] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and multi-domain adaptive allocation algorithms for vnf forwarding graph embedding," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 1, pp. 98–112, 2018.

[15] NGMN Alliance, "5g white paper," *Next generation mobile networks, white paper*, vol. 1, 2015.

[16] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "Vnf placement and resource allocation for the support of vertical services in 5g networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 433–446, 2019.

[17] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware vnf placement and chaining based on a flexible resource allocation approach," in *Proc. 13th IEEE Int. Conf. Netw. Service Manage.*, 2017, pp. 1–7.

[18] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE Conf. Comput. Commun.*, 2009, pp. 783–791.

[19] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding lc-vne algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3648–3661, 2016.

[20] B. Spinnewyn, P. H. Isolani, C. Donato, J. F. Botero, and S. Latré, "Coordinated service composition and embedding of 5g location-constrained network functions," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1488–1502, 2018.

[21] D. Harutyunyan, S. Nashid, B. Raouf, and R. Riggio, "Latency–aware service function chain placement in 5g mobile networks," in *Proc. IEEE Conf. Netw. Softwarization*, 2019, pp. 133–141.

[22] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv preprint arXiv:1209.5145*, 2012.

[23] I. ILOG, "Cplex optimization studio," 2014.

[24] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE Conf. Comput. Commun.*, 1996, pp. 594–602.

[25] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for internet topology," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 770–783, 1997.