Delay-aware VNF Placement and Chaining based on a Flexible Resource Allocation Approach

Abdelhamid Alleg[1], Toufik Ahmed[1], Mohamed Mosbah[1], Roberto Riggio[2], and Raouf Boutaba[3]

[1)] LaBRI (CNRS UMR5800), Univ. Bordeaux, Bordeaux INP, France
[2)] Future Networks (FuN), FBK CREATE-NET, Trento, Italy
[3)] David R. Cheriton School of Computer Science, University of Waterloo

aalleg@labri.fr, tad@labri.fr, mosbah@labri.fr, rriggio@fbk.eu, rboutaba@uwaterloo.ca

***Abstract*— Network Function Virtualization (NFV) is a promising technology that is receiving significant attention in both academia and the industry. NFV paradigm proposes to decouple Network Functions (NFs) from dedicated hardware equipment, offering a better sharing of physical resources and providing more flexibility to network operators. However, in such environment, efficient management mechanisms are crucial to address the problem of Placement and Chaining of Virtual Network Functions (PC-VNF). In this paper, we introduce a PC-VNF model based on a flexible resource allocation approach that takes into account service requirements in terms of latency, in addition to traditional connectivity and resource utilization. This is particularly important for emerging 5G services such as ultra-reliable, low latency and massive machine type communications. The end-to-end performance needs to meet the user expectations as well as service requirements to provide the desired QoS/QoE. Our main goal is to determine the optimal VNF placement minimizing resource consumption while providing specific latency (i.e., end-to-end delay) and avoiding violation of Service Level Agreements (SLA) by constraining allocated resources to a given VNF to reach its required performance. Results show that our approach achieves the required latency with better resources utilization compared to the classical approaches, with a reduction of up to 40% of resource consumption and a higher rate of accepted requests by recovering 15 to 60 % of the rejected requests.**

Keywords*— *Network Function Virtualization; Placement and chaining VNF, QoS (Quality of Service), QoE (Quality of Experience).

I. INTRODUCTION

The majority of network services are built on top of physical proprietary hardware devices known as middle boxes. The diversity and the increasing number of new services requested by users have led to significant CAPEX and OPEX supported by operators to purchase, store and maintain these middle boxes. Recently, network providers started shifting towards virtualized and softwarized network infrastructures capable to offer innovative services to subscribers and keep abreast of continuous changes.

Network Function Virtualization (NFV) has been proposed as a means to meet this need and provide concrete solutions to underlying challenges of placement, management, chaining and orchestration of network services. In particular, NFV provides a malleable approach to design, deploy and manage network services. The main idea is to substitute by software appliances network functions, such as Deep Packet Inspection (DPI), firewalling, media gateways and intrusion detection, that were until now carried-out by dedicated hardware devices. Virtualized Network Functions (VNFs) are easy and inexpensive to deploy, maintain, upgrade and scale over virtualized network infrastructures (NFVI).

Each network service is represented by a Service Function Chain (SFC) and has specific requirements in terms of latency, throughput, and error rate in order to offer a specific QoS (Quality of Service). Next generation 5G networks propose to classify network services into different categories offering diverse performances for broadband communications, mission critical communications, massive IoT, etc. The aim is to slice the network resources (computing and networking) so as to support requirements for diverse services, use cases, and business models ranging from high throughput services to latency-sensitive services.

Each SFC is composed of a sequence of VNF instances that require a specific amount of resources. The allocated resources (computing, memory, storage and bandwidth) and the physical location of each deployed VNF instance will affect considerably the resulting end-to-end QoS performance such as latency.

In this work, we formulate the Placement and Chaining problem of VNFs (PC-VNF) as a Mixed Integer Quadratically Constrained Program (MIQCP) called Flexible Resources Allocation Model (FRAM). The novelty of our work lies in considering the relationship between the resources allocated to a VNF instance and the expected latency, this way ensures delay-awareness in the placement and chaining of VNFs process.

For comparison purpose, we develop a Strict Resources Allocation Model (SRAM) as a baseline that represents exiting approaches which did not consider the aforementioned relationship and solve the PC-VNF by allocating the *exact*

amount of resources requested by VNFs. We define the processing delay or latency as the time needed for a data packet to pass through a VNF instance from ingress to egress.

Existing approaches based on strict resource allocation tackle the PC-VNF problem by answering the following question “How much resources does each VNF require to be instantiated?”. However, our flexible resource allocation approach handles this problem differently by addressing the question of "How much resources have to be allocated to the VNF in order to satisfy the required latency?”. This will provides solutions to the PC-VNF problem that satisfy delay requirements without allocating unnecessarily resources hence avoiding overconsumption of network resources. Moreover, our approach sheds light on new aspects of the PC-VNF problem, such as the variation of performance according to the type of VNF for the same amount of allocated resources and the impact of the VNF implementation on resource consumption.

The rest of this paper is organized as follows. Section II presents relevant related work. Section III discusses the Network Model, the PC-VNF problem and define the resource-delay dependency. Section IV describes both FRAM and SRAM models and their mathematical formulations. Section V evaluates their performance, and finally section VI concludes the paper.

II. Related Work

NFV promises to reduce CAPEX and OPEX for network operators by leveraging commodity hardware as a platform for deploying softwarized network functions. It also provides a more flexible means for designing and managing network services. A large number of research papers have been published on the placement and chaining of VNF (PC-VNF) problem [1] with the goal of optimizing the placement of VNFs subject to different optimization objectives (number of VNF instances, resource utilization, provising cost, etc.). Works in this field can be classified according to their formulations (model and objective) proposed by their solutions.

Addis *et al.* [2] modelled the problem of VNF service chain placement and routing as a Mixed Integer Linear Program (MILP) to optimize both network link and compute resource utilization. Considering two forwarding modes (Standard and Fastpath), they defined the delay introduced by a VNF as function of traffic demands. In the first mode, the processing and forwarding latency is defined as a linear function while in the second, it is defined as constant up to a maximum aggregate bit-rate. However, their model considers VNF delay as a function of traffic load rather than a function of the amount of physical resources allocated to a VNF instance. For example, similar VNFs can have different processing delay when dealing with same traffic demand but using different amount of resource. (the case when varying the number of vCPU). In addition, authors in [7] formulated an ILP model and proposed a heuristic procedure based on a binary search. They associate a processing delay value to each type of VNF but this value is static and constant regardless of the amount of allocated resource or the traffic load.

Mehraghdam *et al*. [5] proposed a mixed-integer quadratically constrained programming (MIQCP) with different optimization objectives for VNF placement. They developed a heuristic to specify the VNF service chain. The MIQCP demonstrated the effect of optimization objective on the obtained VNF placement. However, authors did not consider the delay introduced by VNF instances and its impact on end-to-end delay.

Bari et al. [6], solved the placement problem by finding the required number of VNF instances to optimize OPEX (node and link resource utilization level) by formalizing an ILP. Moreover, heuristics based on dynamic programming are proposed to solve larger networks. As a service level agreement (SLA) constraint, they only considered link propagation delay and did not tackle latency introduced by node processing delay.

Taleb *et al.* [15] considered two competing objectives for VNF placement in mobile core network and proposed three solutions. First, ensuring an acceptable QoE by a near-user placement of data anchor gateway. Second, avoiding the mobility anchor gateway relocation by placing VNFs far enough from users. The third solution is a trade-off between the two previous solutions modeled using game theory [17]. The scope is however limited to only two particular mobile core network functions and VNF resource requirements have not been considered.

Table I. Models addressing the PC-VNF problem

Works	Model	Heurstic	Processing delay	delay consideration	Objective
[2]	MILP	Multi-objective math-heuristic	✓	as function of traffic demands	Minimize number of cores (CPU) and optimize link utilization
[5]	MIQCP	Data rate based algorithm	x	Only link latency	Optimize the latency, the remaining data rate and the number of nodes
[6]	ILP	Dynamic programming	x	Only Propagation delay	Minimize the operational cost of a network.
[7]	ILP	Binary search heuristic	✓	Constant processing delay	Minimize the VNFs number
[8]	ILP	Selective heuristic	x	No delay consideration	Minimize the provisioning cost
[14]	ILP	-	x	No latency consideration	Optimize both Virtual Network Topology and Virtual Network Embedding
[15]	ILP	-	x	Only transfer delay	Minimize the path length and optimize the sessions mobility.

In a similar context, Baumgartner *et al.* [14], investigated the placement of virtual mobile core network functions excluding VNFs on the radio access network. They aimed to minimize resource provising cost while meeting VNF requirements in terms of bandwidth, processing and storage. However, they do not address latency constraint.

Riggio et *al.* [8] examined the VNF placement problem in the radio access network (RAN) domain including functions such as load-balancing, firewall, and virtual radio nodes. An ILP model and a heuristic are proposed. Their objective is to minimize the cost of mapping virtual functions to substrate network (nodes and links) while satisfying VNF requirements in terms of CPU, memory, storage, radio, and bandwidth resources. However, they do not consider the delay introduced by the VNFs. Table I. summarizes some of the most prominent works in the literature.

It is worth noting that, none of the aforementioned works considered the impact of the amount of allocated resources to a given VNF on its processing delay. Indeed, specific resource requirements for each VNF are determined in advance. However, these values can change depending on many parameters such as physical host performance, workload variation, topology changes and utilization level. So, the relationship *allocated resource-VNF performances* should be considered to satisfy SLA requirements with respect to the underlying infrastructure performance and status. We focus in this paper on leveraging this relationship. More details about our approach will be provided in the next section.

III. Network Model and Virtual Function Placement and Chaining Problem

In this section, we provide a definition of the PC-VNF problem and its formulation. Then, we investigate and describe the nature of the dependency relationship between the attributed resources to a given VNF and its performance level in terms of processing delay and its impact on the end-to-end delay. This will allow us to define our model based on a flexible resource allocation approach that takes into consideration the aforementioned dependency.

A. Network model

We adopt a Network Function Virtualization architecture composed of NFVI (physical network), set of VNFs and NFV management and orchestration (MANO) platform, in accordance with the terminology presented in [3]. The NFVI consists of hardware resources including a set of Physical Nodes (PNs) that are able to host a certain number VNFs depending on their resource capacities. PNs are connected via Physical Links (PLs) that forward traffic between VNFs composing a Service Function Chain (SFC). Each SFC can be abstracted as a graph containing VNFs (as nodes) and the network demand between these VNFs (as edges).

Table II. summarizes the NFVI and SFC notation and parameters.

B. NFVI Model

We model the network infrastructures as a graph $G_p(N_p, E_p)$, where N_p is the set of physical nodes (PNs) that compose network and E_p is the set of physical bidirectional links (PLs) connecting these nodes. Each PN $n \in N_i$ represents a possible location that can host a single or multiple VNF instances.

Each PN $n \in N_p$ represents the quantity of available resources in terms of Computing, Memory and Storage denoted Θ^n. Similarly, each PL $(n,m) \in E_p$ connecting two PNs $n,m \in N_p$ has its capacity (Bandwidth, Bitrate, etc.) denoted $\delta^{(n,m)}$.

Table II. NFVI and SFC Notation

Par.	Description
NFVI	
G_p	NFVI graph
N_p	Set of physical nodes in G_p
E_p	Set of physical links between phyical nodes
Θ^n	Available resource at physical node $n \in N_p$
$\delta^{(n,m)}$	Available capacity of physical link $(n,m) \in E_p$
$D_{Tran}^{(n,m)}$	Transmission delay of the physical link $(n,m) \in E_p$
SFC	
G_v	SFCs graph
N_v	Set of VNFs in G_v
E_v	Set of virtual links between VNFs in G_v
N_v^r	Set of VNFs composing the request r where $N_v^r \subseteq N_v$
E_v^r	Set of links between VNFs $\in N_v^r$ such as $E_v^r \subseteq E_f$
$\Omega^{(k,l)}$	Required capacity of virtual link $(k,l) \in E_v^r$
$\Psi^{n'}$	Requested resources of VNF $n' \in N_v^r$
$\Phi_{Alloc}^{n'}$	Allocated resources to VNF $n' \in N_v$
$D_n^{n'}$	Processing delay of VNF $n' \in N_v^r$ mapped into node $n \in N_p$ using an amount of resources equl to $\Phi_{Alloc}^{n'}$
$D_{Proc}^{n'}$	Processing delay generated by VNF $n' \in N_v^r$ using exaclty the required amount of resources $\Psi^{n'}$
D_{th}^r	End-to-end delay threshold associated to $r \subseteq R_{sfc}$
$D_{Max}^{n'}$	Maximum reachebale processing delay of VNF *n'* using the Minimum allowed amount of resources $\Psi_{Min}^{n'}$
$D_{Min}^{n'}$	Minimum reachebale processing delay of VNF *n'* using the Maximum allowed amount of resources $\Psi_{Max}^{n'}$

C. Service Function Chain

SFC defines an ordered (*resp.* partially ordered) set of VNFs. Ordering constraints must be applied to packets, frames, and/or flows obtained after a classification process. The term "service chain" is often used as shorthand for "service function chain"[4].

We note R_{sfc} the set of SFC requests. Each SFC request $r \subseteq R_{sfc}$ is modeled as a subgraph $G_v^r(N_v^r, E_v^r)$ where $N_v^r \subseteq N_v$ is a set of VNFs and $E_v^r \subseteq E_v$ is a set of directed edges called virtual links connecting these VNFs. In addition, each VNF instance $n' \in N_v^r$ has its own requested amount of resources denoted $\Psi^{n'}$. Also, each virtual link $(k,l) \in E_v^r$ connecting two VNFs $k,l \in N_v^r$ has some characterizing metrics (bitrate,

delay, etc.) denoted $\Omega^{(k,l)}$.

Let us define D_{th}^{r} as the target latency, which delimits the end-to-end delay threshold associated to each SFC $r \in R_{sfc}$. The value of D_{th}^{r} is expected to meet a specific requirement according to the type of the deployed service as presented in Table III. We define the end-to-end delay provided by a deployed SFC as the sum of delay processing of its component VNFs instances and the delay needed to forward the flow between these VNFs.

Table III. SERVICE DELAY REQUIREMENTS

Type of Service	Delay end-to-end delay requirment
Conversational Services (CS)	≤ 150 ms
Streaming Services (SS)	≤ 300 ms
Background Services (BS)	≤ 600 ms

D. Resource-Delay Dependency

In the following, we explore an important aspect that was not considered in previous works when addressing PC-VNF problem. Indeed, the fact of handling VNFs that are basically software (programs), allows us to suppose intuitively that the delay needed to execute any VNF is necessarily impacted by the amount of resources allocated to this VNF. There is a large body of literature in the area of parallel and distributed computing community that describes the behavior of a system when increasing its resources [11][12][13].

For the sake of simplicity, we assume that the resource-delay dependency is linear (Formula 1) and that processing delay $D_n^{n'}$ generated by a VNF *n'* mapped into the physical node *n* is defined as a function of allocated resources $\Phi_{Alloc}^{n'}$:

$$D_n^{n'} = f(\Phi_{Alloc}^{n'}) = a_{n'} \cdot \Phi_{Alloc}^{n'} + b_{n'} \quad (1)$$

Where $a_{n'}$ and $b_{n'}$ are given by the formulas (2) and (3):

$$a_{n'} = \frac{D_{Max}^{n'} - D_{Min}^{n'}}{\Psi_{Min}^{n'} - \Psi_{Max}^{n'}} \quad (2)$$

$$b_{n'} = \frac{D_{Min}^{n'} \cdot \Psi_{Min}^{n'} - D_{Max}^{n'} \cdot \Psi_{Max}^{n'}}{\Psi_{Min}^{n'} - \Psi_{Max}^{n'}} \quad (3)$$

We define $D_{Min}^{n'}$ (resp. $D_{Max}^{n'}$) as the minimum (resp. the maximum) processing delay that can be provided by a VNF instance $n' \in N_v$ and it is reached by allocating an amount of resources equal to $\Psi_{Max}^{n'}$ (resp. $\Psi_{Min}^{n'}$). Additionally, $[\Psi_{Min}^{n'}, \Psi_{Max}^{n'}]$ is defined as the *operating range* of VNF *n'*.

The dependency defined in our approach represents an approximation of the well-known Amdahl's law [11] that gives the theoretical speedup in terms of execution time of a fixed workload task as a function of the number of processors executing it. Also, this verifies two important properties which are: *1)* the speedup depends on the parallelizable portion of the program. *2)* beyond a given number of processors the execution time will remain fixed.

The aforementioned properties are preserved by our vision of Linear Dependency as illustrated in Fig 1. Indeed, the processing delay is *specific* to each VNF (software).

Thus, this delay depends on how this VNF is implemented (portion of parallel part of VNF program). The second property is verified by the fact that for any amount of resources higher than $\Psi_{Max}^{n'}$ the delay generated reaches its lower bound $D_{Min}^{n'}$ and cannot be further improved. Moreover, for an amount of resources less than $\Psi_{Min}^{n'}$ the VNF n' cannot be executed.

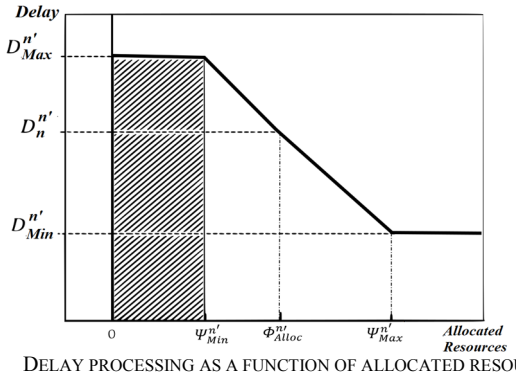

Fig 1. DELAY PROCESSING AS A FUNCTION OF ALLOCATED RESOURCES (LINEAR DEPENDENCY CASE)

IV. PROBLEM FORMULATION

A. Flexible Resources Allocation Model (FRAM)

The problem addressed in this paper can be formulated as a MIQCP model that takes into consideration the aforementioned dependency. The inputs to the placement phase are the network capacity (nodes and links capacities) and latency requirements of different SFC requests. The output represents the optimal solution for placement and chaining of VNFs that minimizes the resources allocation while meeting the delay condition defined by D_{th}^{r}. The objective function that we have formulated is given in (4):

$$Min \left(\sum_{n \in N_p} \sum_{n' \in N_v} (\Phi_{Alloc}^{n'} \cdot \Delta_n^{n'}) + \sum_{(n,m) \in E_p} \sum_{(k,l) \in E_v} (\Omega^{(k,l)} \cdot \Delta_{(n,m)}^{(k,l)}) \right) \quad (4)$$

$\Phi_{Alloc}^{n'}$ is an integer variable indicating the amount of resources allocated to VNF instance $n' \in N_v$ in physical node $n \in N_p$. $\Delta_n^{n'}$ (*resp.* $\Delta_{(n,m)}^{(k,l)}$) is a binary variable indicating whether VNF instance n' (*resp.* virtual link $(k,l) \in E_v$) is mapped onto the PN n (*resp.* onto the physical link $(n,m) \in E_p$). The optimization objective is subject to the following constraints:

$$\sum_{n' \in N_v} (\Phi_{Alloc}^{n'} \cdot \Delta_n^{n'}) \leq \Theta^n \qquad \forall\, n \in N_p \quad (5)$$

$$\sum_{(k,l) \in E_v} \left(\Omega^{(k,l)} \cdot \Delta_{(n,m)}^{(k,l)} \right) \leq \delta^{(n,m)} \qquad \forall\, (n,m) \in E_p \quad (6)$$

$$\Psi_{Max}^{n'} \geq \Phi_{Alloc}^{n'} \geq \Psi_{Min}^{n'} \qquad \forall\, n' \in N_v \quad (7)$$

$$\sum_{n' \in N_v} \sum_{n \in N_p} \left(a_{n'} \cdot W_{Alloc}^{n'} + b_{n'}\right) \cdot B_n^{n'} + \sum_{(n,m) \in E_p} \sum_{(k,l) \in E_v} \left(D_{Tran}^{(n,m)} \cdot \Delta_{(n,m)}^{(k,l)}\right) \leq D_{th}^r \quad \forall r \in R_{sfc} \quad (8)$$

$$\sum_{m \in N_p} \Delta_{(n,m)}^{(k,l)} - \sum_{m \in N_p} \Delta_{(m,n)}^{(k,l)} = \Delta_n^k - \Delta_n^l \quad \forall n \in N_p, \forall (k,l) \in E_v \quad (9)$$

$$\sum_{n \in N_p} \Delta_n^{n'} = 1 \quad \forall n' \in N_v \quad (10)$$

Constraint (5) ensures that the amount of resources allocated to VNFs does not exceed the available resource Θ^n of physical node $n \in N_p$ while constraint (6) ensures that the bandwidth required by the virtual links mapped onto physical link $(n,m) \in E_p$ does not exceed its available capacity $\delta^{(n,m)}$.

We ensure that the amount of resources allocated to each VNF n' is included in its *operating range* by defining constraint (7). Constraint (8) guarantees that the end-to-end delay does not exceed the delay threshold specified by the service. The first part of the equation is a sum of the delay incurred by packet processing in VNFs, while the second part defines the delay incurred by transmitting packets between these VNFs.

Constraint (9) is introduced to enforce the condition that for each virtual link $(k,l) \in E_v$ there must exist a continuous path $(n,m) \in E_p$ allocated between the pair of physical nodes n,m in which VNFs k,l have been mapped.

Constraint (10) states that each VNF n' has to be mapped only once into the physical infrastructure. In other words, the whole amount of resources (Computer, Memory and Storage) allocated to n' must be provided by *exactly* a physical node *n*.

B. Strict Resource Allocation Model (SRAM)

This model is used in this paper as baseline to which our proposed FRAM is compared. Indeed, SRAM behavior reflects most of works in this filed that do not consider the resource-delay dependency and simply satisfy each VNF by allocating the exact requested amount of resources (not more or less). Such strict resource allocation can impact negatively the quality of the placement. For example, a whole SFC request may be rejected due to the fact that one of its VNFs requires two CPUs while only one CPU is available. Whereas the flexibility offered by FRAM allows to accept this SFC by allocating the available CPU to this VNF if and only if the end-to-end delay constraint is respected.

SRAM is formalized as a Mixed Integer Linear Program (MILP). SRAM is obtained by substituting Constraint (8) that explicitly introduces the linear relationship between the allocated resources and the estimated delay, by the constraint (11) that ensures that for each mapped SFC request the end-to-end delay threshold will be respected without any consideration of the aforementioned relationship.

$$\sum_{n \in N_p} \sum_{n' \in N_v} \left(D_{Proc}^{n'} \cdot \Delta_n^{n'}\right) + \sum_{(n,m) \in E_p} \sum_{(k,l) \in E_v} \left(D_{Tran}^{(n,m)} \times \Delta_{(n,m)}^{(k,l)}\right) \leq D_{th}^r \quad \forall r \in R_{sfc} \quad (11)$$

V. Performance Evaluation

In this section, we evaluate the placement performances of the models presented previously using different types of SFCs. All models were implemented using AIMMS Modeling Optimization version 4.3 [9].

All experiments were performed on Windows 8 server with Intel Core i7-3740QM processor with 16GB of memory. All evaluations are repeated 20 times. Also, the confidence intervals are not reported in order to improve readability since it was always smaller than 5%. We first describe the simulation environment and then discuss the used performance evaluation metrics.

A. Simulation environment

In our simulation, we do not explicitly differentiate the type of resource, we just use one single value to represent node resources. The available PNs resources, the available capacities of PLs and the required capacities of virtual links have fixed values. The transmission delay of PLs are based on the study conducted by Choi et al. [16], which characterizes typical average packet delays in Internet Service Provider (ISP) networks.

However, the target delay threshold D_{th}^r is generated randomly according to the type of service provided by SFC request *r* within the main three categories: Conversational Services (CS), Streaming Services (SS) and Background Services (BS). Similarly, each VNF requires an amount of computing, memory and storage capacities which are randomly generated. The processing delay of each VNF is calculated according to its linear parameters (Equations 1, 2 and 3) which are set in a manner to have a processing delay within the range [10,30]. Table II. summarizes the simulation parameters.

Table II. Simulation parameters

Parameters	Value range	
Number of VNF per service	C1,C2	[1, 3]
	C3,C4	[4, 6]
Delay threshold D_{th}^r	(CS)	150 ms
	(SS)	300 ms
	(BS)	600 ms
Available resources at PNs Θ^n	set at 100%	
Requested resource $\Psi^{n'}$	Uniform [1, 5] %	
Required capacity of virtual link $\Omega^{(k,l)}$	set at 1%	
Transmission delay of PL $D_{Tran}^{(n,m)}$	set at 10 ms	
Processing delay $D_n^{n'} = f(\Phi_{Alloc}^{n'})$	Range [10,30] ms	
Available capacity of PL $\delta^{(n,m)}$	set at 100%	

SRAM and FRAM are evaluated using four structural variants of SFC. The first, **C1** is a linear chain composed of a sequence of VNFs between two endpoints. The second, **C2**

consists of a bifurcated (branched) chain using different VNF in each path between two endpoints. The third and fourth variants (**C3** and **C4**) use the same topologies as the ones described previously, but vary in size. In order to evaluate the two formulations (FRAM and SRAM), we use performance metrics adopted in previous works [8][10]. For each model, we measured the average end-to-end delay, the average resource consumption, the accepted requests rate and the average execution time.

B. Simulation Results

First, we analyze the end-to-end delay provided by the two formulations SRAM and FRAM. Fig. 2 depicts the average end-to-end delay that depends on the class of service and the delay measured between endpoints in all experiments. The end-to-end delay is computed as a sum of network function processing delays and transmission delays between endpoints. FRAM yields provides an acceptable end-to-end delay that does not exceed the required delay threshold specific to each request. In fact, our model tries to attain the exact delay threshold while SRAM yields the lowest end-to-end delay per request by consuming extra resources.

In terms of resource consumption presented in Fig. 3, the SRAM provides the worst result compared to FRAM. Such result is due to the strict resource allocation approach applied by this model. Moreover, by assigning the exact amount of resources requested by VNFs, SRAM uses extra computing resources to reduce unnecessarily the end-to-end delay. Indeed, we can notice that VNF demands are oversized which causes an overconsumption of network resources.

Fig. 4 shows the average rate of accepted SFC requests for both models. As expected, FRAM achieves a better rate of accepted SFC requests in all scenarios when SRAM tends to reject SFC requests as their number increases. This early overload of network resources (30 % accepted SFCs for SRAM versus 90% in the case of FRAM) caused by SRAM, is mainly due to the aforementioned overconsumption, unnecessary delay reduction and the lack of flexibility when allocating resources. Moreover, the number of rejected SFCs reflects the number of potential SLA violations. Indeed, relaxing the end-to-end delay threshold constraint will increase the number of accepted SFCs (for both models) but will at the same time, increase SLA violations. In other words, respecting SLA requirements depends on how the consumption of resources is managed. FRAM allows to recover up to 60% of the rejected SFCs.

Fig.5 depicts the average execution time of both solutions for the set of accepted SFC requests. We notice that the execution time increases significantly for both models when dealing with large components (C4 and C3 for SRAM and C4 for FRAM). However, FRAM presents a better performances compared to SRAM solution. For small SFCs both models show similar behavior but the performance gap widens between them when we increase the number of PNs (larger topology) and use larger SFCs (C3 and C4).

Fig. 6 shows the performance of SRAM and FRAM using 5 SFC requests for each class of service and SFCs with various topological structures. We note that the execution time of both models is sensitive to these two parameters. For example, mapping large SFCs such as C4 generates a significant execution time compared to the other variants. Also, placing services with a high delay requirement such as CS and SS, needs an additional time to be achieved compared to services

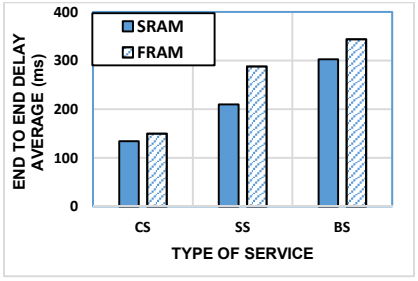

Fig 2. End to end delay for different services

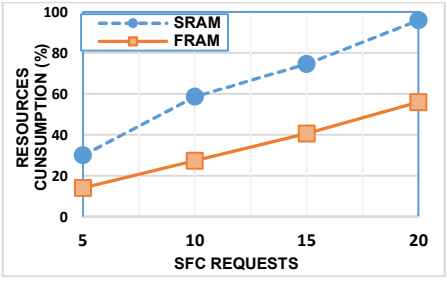

Fig 3. Resource Consumption comparison

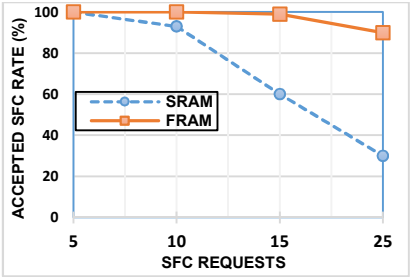

Fig 4. SFC acceptance comparison

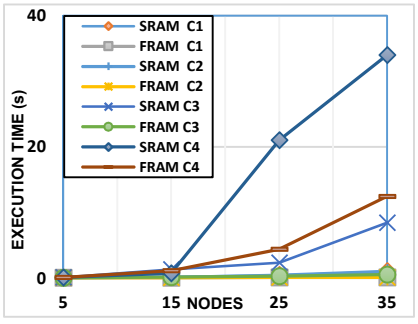

Fig 5. Average execution time comparison

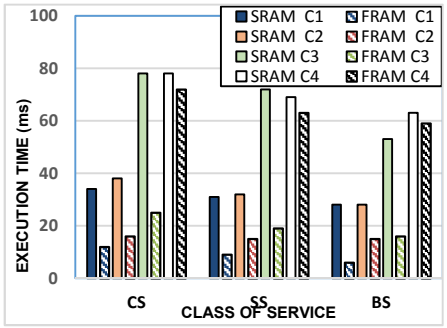

Fig 6. Execution time vs class of service

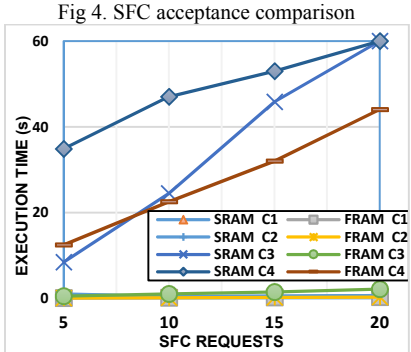

Fig 7. Execution Time vs SFC number

with lower requirements like BS. This extra time is caused by the necessity to find a solution that meets the end-to-end delay threshold. However, the effect of the aforementioned parameters is less apparent in the case of FRAM. This, is mainly due to its ability to gather VNFs composing a SFC on a single PN without having to look for solutions using PLs.

Fig. 7 shows the performance in terms of execution time of SRAM and FRAM using SFCs with various topological structures while increasing the number of SFC requests. We distinguish two patterns. In the first, both models have almost the same performance, the execution time evolves linearly and remains reasonable even for 20 requests (less than 275 ms for FRAM and less than 650 ms for SRAM). In the second pattern, the execution time reaches quickly the time limit of 60s especially for SRAM when using C3 and C4 chains. While the execution time of FRAM evolves linearly when using C4 chains and reaches 40s for 20 requests.

The evaluation of the execution time is motivated by the fact that this metric is vital especially for when designing of an online placement algorithm. In addition, valuable insights and lessons can be drawn on how a SFC should be structured to facilitate and accelerate its deployment. However, in our work the key performance indicators remain the end-to-end delay, the resource consumption and the SFC acceptance rate.

VI. CONCLUSION

NFV is a promising technology as part of network softwarization movement that provides cost-effective mechanism to deploy, operate and maintain network services. In this context, service providers have to address various challenges brought about by the virtualized nature of the network infrastructure while meeting performance expectations in terms of user application requirements.

In this paper, we studied the PC-VNF problem for different applications, particularly focusing on guaranteeing the end-to-end delay requirements. We proposed a Mixed-Integer Quadratically Constrained program (MIQCP) formulation called Flexible Resources Allocation Model (FRAM) that takes into account the Linear Dependency that exists between the amount of resources allocated to a VNF and its processing delay. For comparison purposes, we developed a baseline model that represents existing approaches based on a Strict Resource Allocation approach (SRAM) which ignores the aforementioned dependency.

FRAM results show a better resource utilization compared to SRAM, with a reduction of up to 40% resource consumption and a higher rate of accepted SFC requests by successfully mapping 15 to 60 % of the rejected requests in the baseline approach.

As perspectives for future work, we plan to introduce resources differentiation for both computing and networking resources and to investigate further the relationship between QoS differentiation and performances of SFC. A flexible resource allocation approach will clearly promote the implementation of VNFs in a more parallelizable fashion. Moreover, we aim to devise an alternative heuristic solution for our MIQCP model to handle larger instances of the PC-VNF problem.

REFERENCES

[1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and research challenges", *In IEEE Communications Surveys & Tutorials*, 2015.

[2] B. Addis, D. Belabed, M. Bouet, S. Secci, "VNFs Placement and Routing Optimization". In *Proceedings of IEEE Cloud Networking Conference (CLOUDNET)*, 2015.

[3] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV 003 V1.2.1: Network Functions Virtualization (NFV): Terminology for Main Concepts in NFV", 2014.

[4] P. Quinn and T. Nadeau, "Service Function Chaining Problem Statement," Active Internet-Draft, IETF Secretariat, Internet-Draft draft-ietf-sfc-problem-statement-05, 2014.

[5] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and Placing Chains of VNFs", In *Proc. IEEE 3rd Int. Conf. CloudNet*, 2014.

[6] Bari, M. F., Chowdhury, S. R., Ahmed, R., and Boutaba, R. "On Orchestrating VNFs." *In IEEE 11th International Conference on Network and Service Management CNSM*, 2015.

[7] Luizelli MC, Bays LR, Buriol L, Barcellos MP, Gaspary LP. "Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of VNFs". In *IFIP/IEEE Integrated Network Management Symposium*, 2015.

[8] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, "VNFs Orchestration in Wireless Networks," *In Proc. of IEEE CNSM*, 2015.

[9] J.Bisschop,. AIMMS optimization modeling. Lulu. com, 2006.

[10] M. Barshan, H. Moens, S. Latre, and F. De Turck, "Algorithms for efficient data management of component-based applications in cloud environments", in *Proc. of IEEE NOMS*, 2014.

[11] Amdahl, Gene M. "Validity of the single processor approach to achieving large scale computing capabilities." Proceedings of the April 18-20, 1967, spring joint computer conference. ACM, 1967.

[12] Gustafson, John L. "Reevaluating Amdahl's law." *Communications of the ACM 31.5*, 1988.

[13] McCool, Michael D., Arch D. Robison, and James Reinders. Structured parallel programming: patterns for efficient computation. *Elsevier*, 2012.

[14] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," *in 1st IEEE Conference on Network Softwarization (NETSOFT)*, 2015.

[15] T. Taleb, M. Bagaa, and A. Ksentini. "User mobility-aware virtual network function placement for virtual 5G network infrastructure." In *IEEE International Communications Conference*, 2015.

[16] Choi, B. Y., Moon, S., Zhang, Z. L., Papagiannaki, K., & Diot, C, "Analysis of point-to-point packet delay in an operational network". *Computer networks*, 2007.

[17] Nash, John F. "Equilibrium points in n-person games." *Proceedings of the national academy of sciences* 36.1,1950