SFCaaS: SERVICE FUNCTION CHAINS AS A SERVICE IN NFV ENVIRONMENTS

Tarik Moufakir[1], Mohamed Faten Zhani[1], Abdelouahed Gherbi[1], Moayad Aloqaily[2], Nadir Ghrada[1]
[1]École de Technologie Supérieure (ÉTS), Montreal, Quebec, Canada, [2]Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), UAE

NOTE: Corresponding author: Mohamed Faten Zhani, mfzhani@etsmtl.ca

Abstract – *With the growing deployment of emergent technologies like software-defined networking, network services are expected to be revolutionized. In this paper, we investigate offering Service Function Chains as a Service (SFCaaS) in NFV environments. We describe the potential business model to offer such a service and then we address the service function chain provisioning and resource allocation problem. As the chain is deployed thanks to virtual machines (i.e., instances) and links, we conduct first a detailed study of the costs of Amazon EC2 instances with respect to their location, size, type and performance. Afterwards, we address the resource allocation problem for service function chains from the SFC provider's perspective. We formulate the problem as an integer linear program aiming at reducing operational costs of the service function chains (i.e., costs of virtual machine instances and links, and synchronization among the instances). To address large-scale instances of the problem, we also propose a new heuristic algorithm to reduce operational costs taking into account the conducted study of the costs of Amazon EC2 instances. We show through extensive simulations that the proposed heuristic significantly reduces operational costs compared to a baseline algorithm inspired by the existing literature.*

Keywords – FlexNGIA, network function virtualization, network softwarization, resource allocation, SFCaaS, service function chaining

1. INTRODUCTION

Propelled by emerging technologies like Network Function Virtualization (NFV), Software-Defined Networking (SDN) and data plane programmability, network softwarization has become a new trend aiming at separating software from the networking hardware. This trend is revolutionizing the way networks are designed and managed and opening the door for a new IT open ecosys- tem with new platforms, software and network services and applications [1, 2, 3]. As a matter of fact, network soft- warization bring several benefits including better agility and flexibility, easy automation and faster time-to-market services while ensuring scalability and significantly re- ducing capital and operational expenditure.

Network softwarization allows network/cloud providers (called hereafter SFC provider) to roll out new service offerings. In particular, it would be possible to offer "Service Function Chains" as a Service (SFCaaS) where a whole Service Function Chain (SFC) is offered as a service to a third party (called hereafter service provider). An SFC is an ordered set of Virtual Network Functions (VNFs) that are crossed by packets arriving from the chain's sources and flowing towards the chain's destina- tions (Fig. 1). VNFs are network functions (e.g., firewall, IDS or other functions) implemented in virtual machines or containers running on commodity servers or implemented in dedicated hardware like Field-Programmable Gate Arrays (FPGAs). In this context, the SFC provider would face a major challenge as to how to allocate the resources to the requested SFC in the physical infrastructure. Although this problem has been recently extensively addressed [4, 5, 6] in the literature, we revisit this problem in this work with the following novel contributions:

1. We consider two phases to provision an SFC (Fig. 1). The first one is the translation phase which aims at identifying the optimal number of instances (i.e., vir- tual machines/containers) that are required for each VNF in order to cater to the demand. The second phase addresses the mapping problem, which aims at deciding where to place the instances taking into account the deployment costs. To our knowledge no prior work considered the translation phase.

2. We conduct a detailed study of the costs of Virtual Machines (VMs) (i.e., instances) offered by Amazon EC2 [7] with respect to the location, instance size, and performance. This cost study extends the one carried out by the authors in [8] and sheds light on interesting observations about the costs of instances offered by one of the major cloud providers in the world. This study is used later on in this paper to guide the proposed solutions for the resource allo- cation for the service function chains.

3. We formulate the mapping phase problem as an Integer Linear Program (ILP) aiming at reducing the SFC provider's operational costs of the VM instances and links as well as the synchronization costs among the same-type instances. It is also worth noting that synchronization costs were not considered in existing literature.

4. We also propose two heuristic algorithms to solve the mapping problem with the same aforementioned goals taking into account the conducted study of the costs of Amazon EC2 instances. The first one is a basic and intuitive mapping algorithm (called baseline) and the second one is a more sophisticated algorithm called SFC decomposition-based provisioning (SPIN) algorithm. We show through extensive simulations that SPIN significantly reduces the operational costs compared to the baseline.

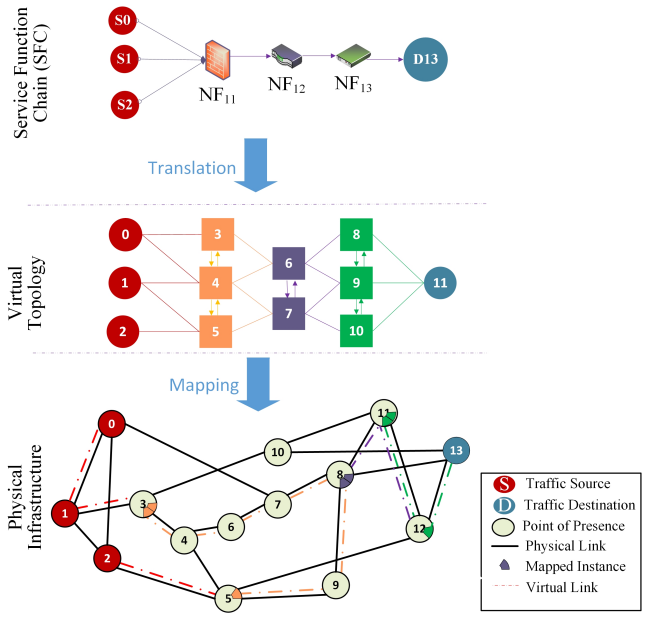

Fig. 1 – SFCaaS - SFC translation and mapping [1]

The remainder of this manuscript is organized as follows. Section 2 describes the SFCaaS service model, highlighting the business model and the potential involved stakeholders as well as the technical challenges related to the deployment of such a service. Section 3 provides an overview of the related literature focusing on service function chaining and highlights the novelty of this work. Section 4 presents the detailed study of the costs of Amazon EC2 instances with respect to several parameters and presents its main outcomes. Section 5 presents the ILP formulation of the mapping problem. Furthermore, Section 6 introduces the two proposed greedy algorithms to address the mapping problem. The experimental results are provided in Section 7. Finally, conclusions and key research directions are described in Section 8.

2. SFCAAS - BUSINESS MODEL, BENEFITS AND CHALLENGES

In this section, we propose the following business model that is adapted to SFCaaS. This model identifies the stakeholders involved in an environment where SFCs are provisioned and offered as a service. We mainly identify three stakeholders defined as follows:

- SFC provider: This is a company that offers "Service Function Chains" as a Service (SFCaaS). It owns and manages a physical infrastructure and is in charge of deploying platforms and software required to run network functions building the chains and also of provisioning and managing the requested SFCs. It should hence perform the SFC translation phase to identify of the number of virtual machines and links needed to build the chain. These components would make up a virtual network as shown in Fig. 1) that should be later on mapped onto the infrastructure. The virtual network is obtained by identifying the number of instances needed to implement each VNF and the virtual links used to connect them (Fig. 1). Note that, in addition to virtual links connecting the VNFs, synchronization virtual links should also be created to connect the instances implementing the same VNF in order to ensure synchronization among them. Indeed, synchronization is needed to guarantee the normal operation of a network function implemented in multiple instances (e.g., IDS) [9].

- Service provider: This could be a company or institution that has users spread around the world. A service provider needs to define the SFC needed to run its service, its composition, performance requirements, and identifies the chain sources/destinations. The composition of the SFC refers to the type of each network function (NF) making up the chain. The performance requirements could be in terms of end-to-end delay, packet loss, traffic demand, resources (CPU, memory and disk) and other parameters. Of course, the service provider relies on the SFC provider to provision the SFC and allocate the needed resources.

- User: These are customers of the service provider and are located at the sources or destinations of the service function chain. The traffic coming from users will be steered across the SFC provisioned by the service provider.

Potential SFC providers could be major companies offering cloud services like Google, Amazon EC2, and Microsoft that have their own network with predictable performance and has computing resources spread across the network [10, 11] . For instance, the AWS infrastructure shown in Fig. 2 is a software-defined worldwide global infrastructure [10] with 25 regions across the world serving 245 countries and territories where each region contains one or more data centers. The infrastructure has also more than 218 edge locations and 12 regional edge caches for a total of 230 points of presence. Regions are connected through a private world-wide network managed by Amazon AWS, which makes it easy for Amazon to predict and even fully control the network performance. As computing resources are available everywhere in this global infrastructure and such companies have the expertise on technologies like cloud com-

puting, virtualization, and SDN, it is straightforward that they could provide SFCs as a service and easily provision and allocate the required resources across their global infrastructure.

Fig. 2 – AWS global infrastructure - source [10]

• **SFC benefits:** Similar to traditional cloud computing services, offering SFCs as service would bring several benefits for service providers, such as avoiding management hassle with no need to software and hardware maintenance. In addition, costs will be reduced for service providers as there is no capital and operational expenditure. It would also be possible to get low prices thanks to the economies of scale. Furthermore, SFCaaS would allow predictable performance thanks to the SFC provider's expertise and knowledge of the infrastructure (i.e., topologies, characteristics, performance). SFC providers could also offer novel and customized network functions that are carefully implemented and managed to offer optimal performance [1].

• **Technical challenges:** From the SFC provider's perspective, the main challenge is to provision SFCs while maximizing profit, minimizing operational costs and satisfying the SFC requirements in terms of end-to-end delay. In this context, we can identify several challenges that can be summarized as follows:

- Decide how many instances to use implementing a VNF: Implementing a VNF in a single instance (VM/Container) may not be sufficient for several reasons. Indeed, a single instance may not have enough resources to handle the incoming traffic. Hence, implementing the same VNF in multiple instances allows us to overcome the lack of resources by distributing the function over several Points-of-Presence (PoPs). It also allows us to reduce delays, costs and to improve fault tolerance as the failure of an instance would not necessarily affect the others. At the opposite side, when a VNF is implemented in multiple instances, there might be some drawbacks. For instance, some data might need to be synchronized among the different instances in order to ensure a normal operation of the network function (e.g., distributed intrusion detection systems [9]). As a result, one needs to consider the cost of synchronization among the same-type instances. This cost can be expressed in terms of CPU, memory and bandwidth consumed in order to ensure synchronization. There might also be constraints on the delay needed to carry out the synchronization [9]. In this context, deciding how many instances are needed to implement one VNF and what are the synchronization costs and constraints are key challenges when provisioning the SFC.

- Decide the type of VM instance to use to run the VNF: The selection of the VM instance type depends on the network function requirements (in terms of resources like vCPU, memory, storage), processing capacity (packets per second), and the operational costs of running the instance. Of course, the decision should take into account the network function properties (the nature of the function itself, the used software and operating system, database and other software) as well as the geographical location of the instance which has an impact on the cost and access delay.

- Allocate resources for the chain: The third challenge is to identify where to allocate the resources to place the instances in the physical infrastructure and how to allocate the bandwidth for the virtual links to connect instances. The goal would be to maximize the SFC provider's profits and minimize its operational costs.

In this work, we try to approach the aforementioned challenges and study the parameters and considerations to address them. In the following section, we summarize the existing literature that has attempted to address the same problems.

3. RELATED WORK

In this section, we briefly present recent research work addressing the provisioning problem of service function chaining, i.e., the problem of placing and chaining of a set of ordered VNFs. In recent years, a large body of work has studied this problem [12, 13, 14, 15, 9]. Proposed solutions have attempted to reach several objectives like minimizing operational costs [16, 17, 18], minimizing network utilization [16, 19], minimizing latency [20, 21, 22, 23, 24, 25, 26] and minimizing resource consumption [27, 28, 29, 22, 24, 30, 26, 31]. The analysis of existing literature on SFC placement and chaining carried out by Santos et al. [32] finds that minimizing operational costs is the most widely sought-after goal as it was the aim of more than 42% of published articles pertaining to service function chaining.

For instance, Carpio et al. [14] formulate the placement and chaining of VNFs as a mixed linear program and compare it with a random fit placement algorithm. For better

scalability, they devise a genetic algorithm to find a sub-optimal solution. The proposed algorithm focuses first on finding admissible paths and calculating link costs then allocating, within those paths, the resources for VNFs. However, the algorithm assumes that the number of VNF instances is known beforehand.

Bari et al. [33] designed a platform to carry out the placement, the chain composition and the monitoring of VNFs. However, the proposed method neither addresses the case where multiple instances are needed nor determines the appropriate placement of VNFs in the infrastructure.

Beck and Botero [34] looked also at the VNF placement and chaining problem and proposed CoordVNF, a solution that aims at minimizing link utilization over the infrastructure. This proposal considered the use of multiple instances of the virtualized deep packet inspection function where the traffic is split into TCP and non-TCP traffic and managed by different instances. However, the costs of instances and their placement were not discussed in this proposal.

Wang et al. [15] address the problem of online deployment of multiple VNF instances in order to process the fluctuating traffic rate received at the VNFs with the goal of minimizing the costs of the provisioned resources. The authors proposed two algorithms, one for provisioning a single service chain and the other for provisioning simultaneously multiple service chains.

Ghaznavi et al. [12] address the VNF placement problem with the goal of reducing server and bandwidth consumption. They introduce a solution to optimize the placement of VNFs by minimizing installation, transportation, reassignment and migration costs of VNF instances. However, the solution assumes that all instances of the chain are of the same type.

Mingshu et al. [35] propose a resource allocation algorithm with the goal of maximizing network resource utilization. This algorithm allocates the shortest and the least costly underlying path to map each of the links of the the service function chain.

Masahiro and Takanori [36] formulate the shortest path tour problem as an ILP to solve the service chaining and function placement and to find the service path that would minimize the delay.

Wang et al. [37] focused on determining paths for the flows that should cross the service function chain while respecting the right order of the network functions. They leverage the concept of virtual layered graph to consider NFV processing latency and use shortest path algorithms to solve the problem.

Chao et al. [38] proposed a mechanism to dynamically deploy network functions. Their approach is based on ant colony optimization and relies on cooperation between multiple forwarding equipment on the packet transmission path to jointly use already placed network functions and further optimize packet delays.

Pham [39] explored the joint optimization of VNF placement and routing in order to maximize cost-efficiency under the delay-guarantee constraint. He formulated the problem as a mixed-integer linear programming model and proposed an algorithm based on reinforcement learning to find an approximation solution for large-scale instances of the problem.

Unlike previous work, in this paper, we consider not only SFC mapping but also the translation phase where the number of instances for each VNF is estimated and considered in order to build a virtual network to be mapped. We also address SFC mapping taking into account synchronization and deployment costs of the VNF instances. Our work relies on realistic data and a study based on the Amazon EC2 instance costs.

In the following section, we start first by studying the costs of the offered general-purpose instances as this study will guide the development of the proposed solutions for the service function chain translation and mapping phases.

4. STUDY OF THE COSTS OF AMAZON EC2 INSTANCES

In this section, we study of the costs (prices) of the instances (i.e., VM) offered by Amazon EC2 with respect to the amount of resources, location, and performance. The outcome of this study is leveraged later while develop- ing our SFC translation and mapping solutions to care- fully place the instances in the physical infrastructure. In particular, we considered Amazon EC2 general-purpose virtual machine instances of type T2 [7] which provide a large range of flavors as shown in Fig 3. These general- purpose instances offer compute, memory and network- ing resources that can be used for diverse types of work- loads including network functions. Each instance flavor defines the amount of vCPU and memory of the virtual machine and has a different cost. The table shows the hardware characteristics on which the VM flavor would run according to Amazon (Fig. 3). In the following, we study several aspects related to the cost of these flavors and their performance. The study includes an analysis of the instance cost versus its location, its software stack, and its allocated resource amount, and, finally, the VNF performance (i.e., packet processing capacity) versus the instance type.

• **Instance cost vs. location:** Fig. 4 shows the instance price for 15 locations in the Amazon infrastructure. It is clear that instance costs significantly vary from one location to another. According to the figure, the difference in cost between two locations for the same instance type can go from 0.01$ and can reach 1$ for large instances. It is worth noting that even a small difference in the instance cost may lead to a high impact in the instance cost. For instance, 0.1$/hour cost difference would translate into 86 million dollars a year considering 100K instances, and into around 2 billion dollars for around 2 million instances, which is a lower-bound estimation of the number of instances running on the Amazon EC2 infrastructure [40].

Instance Type	vCPU Cores	Memory (GiB)	Physical machine Processor
t2.micro	1	1	High frequency Intel Xeon processors
t2.small	1	2	
t2.medium	2	4	
t2.large	2	8	
t2.xlarge	4	16	
t2.2xlarge	8	32	
m4.4xlarge	16	64	2.3 GHz Intel Xeon® E5-2686 v4
m4.10xlarge	40	160	
m4.16xlarge	64	256	2.4 GHz Intel Xeon® E5-2676 v3

Fig. 3 – EC2 general-purpose instances [7]

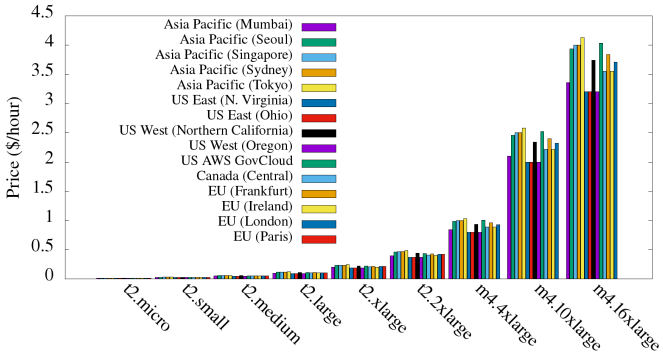

Fig. 4 – Instance price for different locations (note that M4 instances are not available in Paris)

• **Cost vs. software stack:** Fig. 5 shows the price of different instances with different software stack. It considers only the instances located at Amazon AWS Oregon re- gion. The figure shows that instance costs vary depending on the software stack. Linux distributions (e.g., Linux, RHEL, SLES) have similar costs and are much less expen- sive than instances running Microsoft Windows. Further- more, adding additional software to the instance (e.g., SQL Web) would significantly increase the price (e.g., up to 1.5$ for large instances).

• **Instance size vs. cost**: The instance size refers to the amount of resource in terms of CPU and memory that an instance has. We hence aim at evaluating, for the same cost, how much resources we can provision when we use micro-instances (i.e., the smallest instance that has only 1 vCPU and 1GiB of memory) versus larger instances. To do so, Fig. 6 shows the price of all AWS instances and the amount of resources they provide. It also shows how many t2.micro instances could be provisioned for the same price. For instance, as shown in the figure, the price of one m3.16xlarge instance (containing 64 vCPU and 256 GiB of memory) is 3.2 $/hour. For almost the same price, one

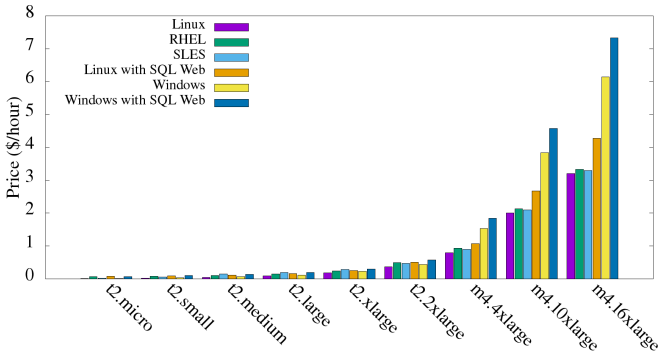

Fig. 5 – Instance price for different software stacks (Oregon)

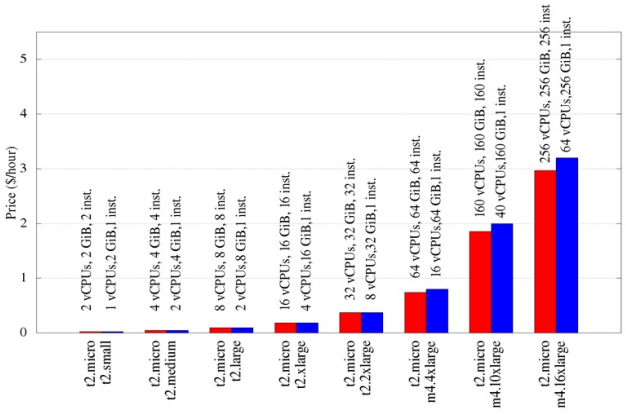

Fig. 6 – Price and size of Amazon EC2 instances compared to t2.tiny instances (US West, Oregon)

could provision 256 t2.micro instances offering 256 vCPU and 256 GiB of memory. This means that if provision 256 t2.micro services, we can get 192 (i.e., 256-64) more vCPUs with the same amount of memory (256 GiB) com- pared to a single m3.16xlarge instance. The same note applies for the other types of instances. As a result, we can conclude that small instances are more cost-effective compared to large instances as, for the same cost, micro-instances would provide roughly four times more vCPUs. This is, of course, interesting if the function/application could run normally in a distributed manner on several instances.

• **VNF performance vs. instance type:** In this experiment, we try to evaluate the packet processing capacity of each instance, i.e., how much packets an instance could process when running a specific VNF. To do so, we conduct experiments using different VNF types running on different instances while gradually increasing the packet arrival rate in order to assess the limit of the instance processing capacity. We assume that the processing capacity of the instance is reached when the CPU utilization of the instance reaches 90% and the packet loss reaches 10%. For instance, Fig 7 shows how the utilization and the packet loss ratio evolution while increasing the incoming packet rate for an Amazon Ec2 instance of type t2.micro running a firewall (Shorewall firewall [41]). We can see

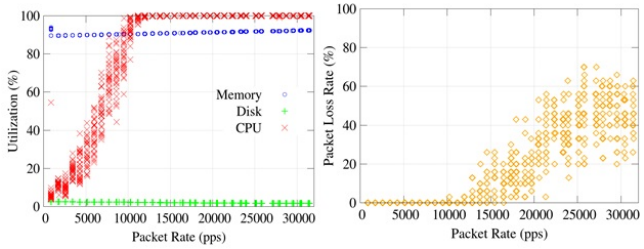

Fig. 7 – Firewall [software: Shorewall, instance: t2.micro]

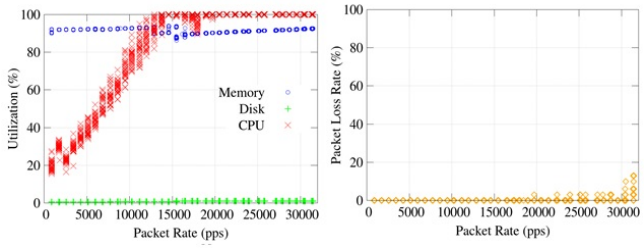

Fig. 8 – IDS [software: Snort, instance: t2.micro]

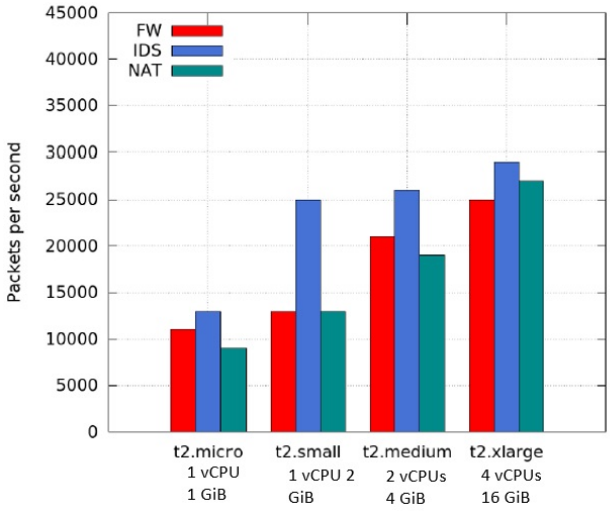

Fig. 9 – VNF processing capacity per instance type

that the CPU utilization reaches 90% and we start having packet loss when the incoming packet rate is around 10,000 packets per second (pps). This means that the processing capacity for this particular network function (Shorewall firewall) on a t2.micro-instance is around 10,000 pps.

We have also conducted the same experiment while running the Snort Intrusion Detection System (IDS) [42] on the same type of instance. The results are reported in Fig. 8 and show that the processing capacity of the t2.micro-instance running the Snort IDS function is 13000 pps.

Fig. 9 summarize the results for three types of network functions, namely a firewall, an IDS and a NAT, that are running on different types of instances. We can clearly see in the figure that, for the same instance type, the packet processing capacity varies from one type of network function to another. Moreover, we can also notice that the processing capacity is not always proportional to the amount of allocated resources. Indeed, we can see in the figure that the t2.xlarge instance has four times the resources than the t2.micro-instance but is not able to process four times the amount of packets processed by the t2.micro. As a result, four micro-instances would process much more packets than a single t2.xlarge instance. While it is not possible to provide a straightforward explanation of this result (as we do not have access to internal statistics of the AWS infrastructure), the reasons for such a result might be the network bottlenecks and also the heterogeneity of physical machines on which the instances are running (see Fig. 3).

The above observation means that distributing a function over multiple small instances would allow higher packet processing capacity and also a lower cost according to the comparison reported in Fig. 6.

• **Study outcomes:** We can summarize the study outcomes as follows:

- Instance costs vary significantly from one location to another.
- The software stack has a big impact on the instance cost.
- The VNF processing capacity is not necessarily proportional to the amount of resources.
- The VNF processing capacity varies significantly from one function to another.
- Small instances are more cost-effective and hence, if there is no synchronization cost, multiple instance deployment is more cost-effective and provides higher processing capacity.

Taking into consideration the above outcome, it is of utmost importance to develop SFC provisioning solutions that are able to find the best trade-off between cost (including instance price, synchronization and bandwidth costs) and processing capacity. In the following, we propose an integer linear program to solve the SFC mapping phase and two greedy solutions to deal with large-scale instances of the problem.

5. MAPPING PHASE: PROBLEM FORMULATION

In this section, we formulate the SFC mapping problem as an Integer Linear Program (ILP) with the objective of minimizing the SFC provider's operational costs in terms of instance deployment costs, bandwidth and synchronization costs.

Table 1 – Table of notations

Symbol	Definition
$G = (N, P)$	Graph G where N is the set of nodes and P is set of physical links
$V = (I, L)$	The virtual network Graph V with I is the set of VNF instances and L is the set of virtual links
C_n	Available capacity at PoP $n \in N$ expressed in number of instances
B_{mn}	Bandwidth capacity of the physical link connecting nodes m and n
$b_{i,j}$	Bandwidth requirement of the virtual link connecting instances i and j
δ_{im}	Deployment costs per unit of time for VNF instance i into PoP m
$\Delta_{m,n}$	Bandwidth cost per bandwidth unit in physical link (m, n)
f_{im}	Boolean constant set to 1 if VNF instance i has to be embedded into node m
s_{ij}	Boolean constant set to 1 if there is a synchronization between instances i and j
x_{im}	Boolean decision variable indicating whether or not instance i is embedded into node m
$y_{ij,mn}$	Boolean decision variable indicating whether virtual link (i, j) is mapped into physical link (m, n)
$\mathbb{C}$	Operational costs
$\mathbb{S}$	Synchronization costs

The physical infrastructure owned by the SFC provider is made from several PoPs that are geographically distributed. The infrastructure is modeled by a graph $G = (N, P)$ where $N = \{0, 1, ..., |N|\}$ represents the set of PoPs and $P = \{(m, n) \in (N \times N) \mid m$ and n are directly connected$\}$ denotes the set of physical links that connect the PoPs. Each PoP $n \in N$ contains an amount of physical resources C_n expressed as the maximal number of t2.tiny instances that the PoP can host. Note that a t2.tiny instance contains 1 vCPU, 1 GiB and 1 GB of memory and disk, respectively. A physical link $(m, n) \in P$ that connects the PoP m with PoP n has a bandwidth capacity B_{mn}.

Furthermore, a service function chain is represented as a graph $V = (I, L)$ where $I = \{0, 1, ..., |I|\}$ is the set of virtual instances in the chain and L is the set of virtual links connecting them.

Each VNF instance $i \in I$ has a resource requirement of 1 vCPU, 1 GiB of memory, and 1GB of storage. Each virtual link $(i, j) \in L$ has bandwidth requirement b_{ij}. It is worth noting that, for simplicity, the endpoints of the chain (i.e., sources and destinations) are also considered instances with requested resources equal to zero. They are con- strained to be mapped onto particular physical PoPs that are provided in the VNF request.

Furthermore, we define two decision variables. The first one is denoted as $x_{im} \in \{0, 1\}$ and indicates whether or not VNF instance i is embedded into PoP m.

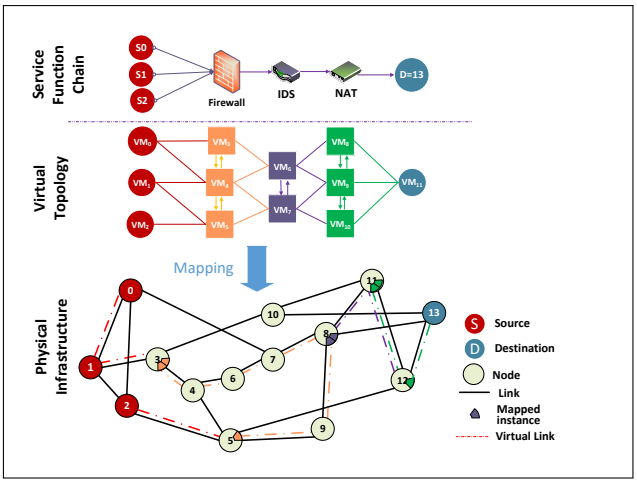

Fig. 10 – SFC embedding problem

The second decision variable is denoted as $y_{ij,mn} \in \{0, 1\}$. If $y_{ij,mn} = 1$, the virtual link (i, j) uses the physical link mn. It is worth noting that a virtual link is embedded through a physical path (i.e, multiple connected physical links). Hence, several physical links could be used to embed a virtual link. In other words, if $y_{ij,mn} = 1$, the physical link (m, n) is part of the physical path used to embed the virtual link (i, j).

• **Objective function:** The objective function when embedding an SFC request aims to minimize the operational costs $\mathbb{C}$ and synchronization cost of the embedded VNF instances $\mathbb{S}$. It can be expressed as:

$$J = \min_{\substack{(x_{im})_{i\in I, m\in N} \\ (y_{ij,mn})_{(i,j)\in L,(m,n)\in P}}} (\mathbb{C} + \mathbb{S}) \qquad (1)$$

In the following, we provide more details on how to compute the operational and synchronization costs:

• **Synchronization cost:** The synchronization cost can be expressed as follows:

$$\mathbb{S} = \sum_{(i,j)\in L} \sum_{(m,n)\in P} y_{ij,mn} s_{ij}\, b_{ij}\, \Delta_{mn} \qquad (2)$$

where $y_{ij,mn}$ indicates whether or not the physical link (m, n) is used for embedding the virtual link (i, j). The Boolean variable s_{ij} is equal to 1 if instances i and j implement the same VNF type and hence require synchronization among them to operate. In this case, there is a synchronization cost computed as $b_{ij}\, \Delta_{mn}$, which is the cost of using bandwidth needed to exchange synchronization data between i and j.

• **Instance and link operational costs:** This is the cost of running the VNF instances on the infrastructure and the bandwidth consumed by the virtual links connecting them. It can be expressed as follows:

$$\mathbb{C} = \sum_{m\in M} \sum_{i\in I} x_{im}\, \delta_{im} + \sum_{(i,j)\in L} \sum_{(m,n)\in P} y_{ij,mn}\, (1 - s_{ij})\, b_{ij}\, \Delta_{m,n} \qquad (3)$$

where δ_{im} is the deployment cost (expressed in dollars per unit of time) of VNF instance i into PoP m. It is worth noting that δ_{im} varies from one PoP to another as it depends on several factors including the electricity price in the PoP, the type of VNF, the license, and the operating system as suggested by the conducted Amazon EC2 study. The first term of the equation (Eq. *(3)*) represents the total cost of deploying the VNF instances. The second term of the operational costs is the total cost of bandwidth consumed by the virtual links. $\Delta_{m,n}$ denotes the cost in dollars (per bandwidth unit and unit of time) for the physical link (m, n). The above objective function is subject to the following set of constraints:

• **SFC endpoints embedding constraint:** SFC endpoints (the sources and the destinations) should be embedded into specific PoPs stated in the request. We define the Boolean variable f_{im} (provided as an input to the ILP) that is equal 1 when the instance i is an endpoint that has to be embedded in PoP m. The following equation captures this constraint:

$$x_{im} \geq f_{im} \quad \forall m \in N, \forall i \in I \tag{4}$$

Instance embedding constraint: This constraint ensures that each VNF instance i is embedded once and only once. It can be expressed as:

$$\sum_{m \in N} x_{im} = 1 \qquad \forall i \in I \tag{5}$$

• **Resource capacity constraint:** This constraint ensures that any hosting PoP has enough resources to host the VNF instances.

$$\sum_{i \in I} x_{im} \leq C_m \qquad \forall m \in N \tag{6}$$

where C_m represents the available capacity at PoP m.

• **Bandwidth constraint:** We must also ensure that the bandwidth capacity required to embed all virtual links in a physical link does not exceed its available bandwidth. This can be expressed as follows:

$$\sum_{i,j \in L} y_{ij,mn}\, b_{ij} \leq B_{mn} \qquad \forall (m,n) \in P \tag{7}$$

• **Flow conservation constraint:** We must also ensure that the incoming traffic to a physical node is equal to its outgoing traffic unless this PoP is a source or a destination. This constraint can be expressed as:

$$\sum_{(n,m) \in P} \sum_{(i,j) \in L} y_{ij,nm}\, b_{ij} - \sum_{(i,j) \in L} x_{jm}\, b_{ij} = \sum_{(m,n) \in P} \sum_{(i,j) \in L} y_{ij,mn}\, b_{ij} - \sum_{(i,j) \in L} x_{im}\, b_{ij} \quad \forall m \in N \tag{8}$$

The service chain embedding problem is an *NP*-hard problem as it generalizes bin-packing problem; therefore, finding an optimal solution is not viable due to the large number of requests processed in the production environment. Hence, we propose two heuristics in the following section to solve this problem and explore potential solutions.

6. MAPPING PHASE - PROPOSED SOLUTIONS

In this section, we address the *NP*-hardness of the problem by putting forward two heuristics solutions, a baseline algorithm and a more sophisticated algorithm called SFC decomposition-based provisioning (SPIN). Both solutions assume multiple sources and a single destination to simplify the problem and aim at minimizing SFC provider's operational and synchronization costs while ensuring that accepted requests satisfy their end-to-end delay requirement. In the following, we provide more details about the two algorithms.

6.1 Solution 1: Baseline algorithm

The baseline algorithm is an intuitive algorithm that aims to satisfy the requirements of the SFC in terms of resources (e.g., CPU, memory, bandwidth) and end-to-end delay while minimizing instance costs. The algorithm proceeds with the following steps. The first step is to estimate the number of instances and virtual links required for the whole chain. The number of instances is simply equal to the number of t2.micro instances needed to process the arriving packet rate. The processing capacity of a t2.micro instance is estimated using the technique described in Section 4 (e.g., Fig. 7 and Fig. 8). Once the number of instances for each VNF is estimated, the virtual topology is built. The second step is to allocate resources for this virtual topology as shown in Algorithm 1. For each source instance of the virtual topology, we start by embedding the virtual nodes (i.e., instances) connected to it (i.e., neighbors). For each of these instances, we recursively embed its neighbors by calling recursively the function $EmbedNeighbors(instance\ i)$ (Algorithm 2). The complexity of this recursive algorithm is $O(|I|^2)$ where $|I|$ is the number of virtual instances in the virtual topology.

Algorithm 1 Baseline

Input: Virtual topology V= (I,L)
Input: Placement constraint $(f_{im})_{i \in I, m \in N}$
Input: Virtual Topology Destination $d \in N$
Output: Boolean Embedded
for all $i \in I$ such that i is a source (i.e., $\sum_{m \in N} f_{im} = 1$) **do**
  $s \Leftarrow$ the hosting physical node of source instance i (i.e.,$f_{is} = 1$)
  $x_{is} \Leftarrow 1$
  Return EmbedNeighbors(i)
end for

Algorithm 2 EmbedNeighbors(instance i)

$s \Leftarrow$ Physical node hosting instance i
for all $j \in neighbors(i)$ (Embedding instances connected to i) **do**
  if j is not embedded **then**
    Find m such that $m \in ShortestPath(s,d)$ & $C_m \geq 1$ & $PathBandwidth(s,m) \geq b_{i,j}$
    if m exists **then**
      $x_{jm} \Leftarrow 1$ (Embed j in m)
      $y_{ij,sm} \Leftarrow 1$ (Embed virtual link (i,j) in physical path (s,m))
      $C_m \Leftarrow C_m - 1$ (Update the node capacity)
    else
      Return False (Instance j is not embeddable)
    end if
  end if
end for
for all $j \in neighbors(s)$ **do**
  if j is not embedded **then**
    Return EmbedNeighbors(j) (Embedding instances connected to j)
  end if
end for
Return True (all instances were embedded)

6.2 Solution 2: SFC decomposition-based provisioning (SPIN) algorithm

This algorithm is called SFC decomposition-based provisioning (SPIN) and proceeds into four phases (Algorithm 3). In the first phase, we estimate the number of instances for each VNF and estimate the number of virtual links just like the way it is done by the baseline algorithm. The second phase is the decomposition phase where the virtual topology is divided into subchains where each subchain is a chain of VNF instances that contains a single instance of each VNF type and connects one source to one destination.

The third phase is the subchain embedding phase (Algorithm 4) where each subchain is embedded in the shortest path between the source and destination of the subchain denoted as P. The path P is selected as the one with the lowest cost and that has a delay satisfying the e2e delay requirement of the chain and has enough resources to em- bed the subchain ($FreeInst(P)$ is the number of free in- stances in the path P and $NumberInstances(SC_k)$ is the number of instances needed by subchain SC_k) . The vir- tual links intended to carry the synchronization traffic are then provisioned between the same-type VNF instances (Function $EmbedSynchVirtualLinks(V)$) . The last phase is the optimization phase 5 that consists of selecting each instance and explores the possibility of migrating it in one of the physical nodes that are neighboring its current physical location. The goal is to further reduce operational and synchronization costs (Eq. *(1)*) while always ensuring that the requested bandwidth and e2e delay is satisfied.

The complexity of SPIN algorithm is $O(K)$ where K is the number of subchains. The complexity of the optimization phase is $O(|V|)$ where V is the number of virtual instances in the virtual topology.

Algorithm 3 SPIN

Input: Virtual topology $V = (I, L)$
Input: Placement constraint $(f_{im})_{i \in I, m \in N}$
Input: Virtual Topology Destination $d \in N$
Output: Boolean $Embedded, VLEmbedded$
Decompose V into K subchains $(SC_k)_{(k=1..K)}$
repeat
  $Embedded \Leftarrow EmbedSubchain(SC_k)$
  $k \Leftarrow k + 1$
until ($Embedded = False \parallel k = K + 1$)
$VLEmbedded \Leftarrow EmbedSynchVirtualLinks(V)$
if ($Embedded$ & $VLEmbedded$)=True (Embedding is successful) **then**
  Optimization(V) (optimization phase)
  Return $True$
else
  Return $False$
end if

Algorithm 4 EmbedSubchain(subchain SC_k)

$P \Leftarrow$ Find path with minimal cost such that $delay(P) \leq delay(SC_k$ & $FreeInst(P) \leq NumberInstances(SC_k)$ & $Bandwidth(P) \geq Bandwidth(SC_k)$
if P exists **then**
  Embed SC_k in P
  Return True (SC_k is successfully embedded)
else
  Return False (SC_k is not embeddable)
end if

7. PERFORMANCE EVALUATION

7.1 Simulation setup

In order to evaluate the performance of the proposed algorithms, we developed a C-based simulator that simulates the physical topology and carries out the translation and mapping of the SFC requests. Each simulation assumes the arrival of requests during two months.

The physical infrastructure is assumed to contain 25 nodes with each node having a hosting capacity randomly set between 50 and 100 t2.micro instances. The nodes are connected with 10 Gbps links with propagation delays randomly set between 10 and 50 ms.

The SFCs were generated randomly with an average arrival rate set between 0.1 and 0.15 rps following a Poisson distribution. The average lifetime of the requests follows an exponential distribution with an average of 1 hour. The average number of VNFs per SFC is 10 with an average number of sources around 7. The demand in terms of packet arrival for each SFC is generated randomly between 2000 and 120 000 packets per seconds.

Algorithm 5 Optimization(VirtualTopology V)

for all $i \in V$ (Parse all instances) **do**
  $n \Leftarrow$ Physical node hosting i
  for all $m \in neighbors(n)$ (Explore migrating i to neighboring nodes) **do**
    $Cost \Leftarrow CurrentVCost(V)$ (Compute Embedding Cost Eq. *(1)*)
    $NewCost \Leftarrow VCost(V, i, m)$ (Compute cost assuming i is hosted in m)
    if $CheckConstraints(V)$ & $NewCost < Cost$ (all resource constraints should be satisfied) **then**
      Migrate(i, m) (migrate instance i to physical node m
    end if
  end for
end for

The end-to-end delay requirement of an SFC request is computed as follows: $\max_{s,d}(t_{s,d}) \times$ 130% where $t_{s,d}$ is the path latency between a source s and a destination d where s and d are a source and a destination of the SFC request. This ensures that, theoretically, the end-to-end delay requirement between the sources of the SFC and its destination could be satisfied as it is 30% higher than any path between the sources and the destinations of the request.

We also assume that we have nine types of VNFs. The packet processing capacity of each type of VNF is generated randomly between 2000 to 12000 packets per second (pps) when running on a t2.micro-instance. The synchronization cost among same-type instances is 0.01$/hour multiplied by the type of the instance. We used the Amazon EC2 instance prices as instance costs. The instance revenue as the cost of the instance plus 0.1$/hour. this means that there is 0.1$/hour profit for the SFC provider each instance. In the next subsection, we present the results generated for the two proposed algorithms under the abovedescribed simulation setup.

7.2 Simulation results

We first compare the performance of the baseline and SPIN algorithms for an arrival rate 0.03 requests per second (rps).
As shown in figures 11 and 12, SPIN maps 25% more requests and leads to around 37% higher CPU utilization in the whole infrastructure.
To further assess the performance of the two proposed greedy algorithms for different scenarios, we computed the following metrics while varying the SFC requests arriving rate:

- Acceptance ratio: This is computed as the ratio of the number of accepted SFCs to the total number of received SFC requests. Accepted requests refers to the ones for which the algorithm succeeded in finding enough resources for the SFC while satisfying its e2e delay requirements.

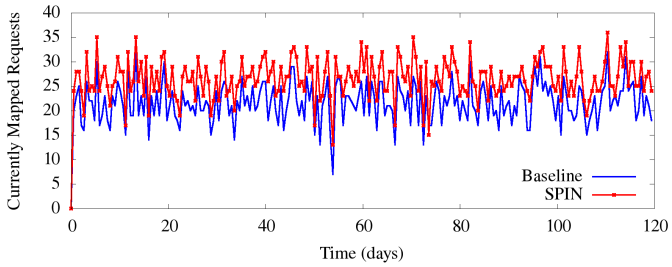

Fig. 11 – Number of mapped requests over time (request arrival rate: 0.03 rps)

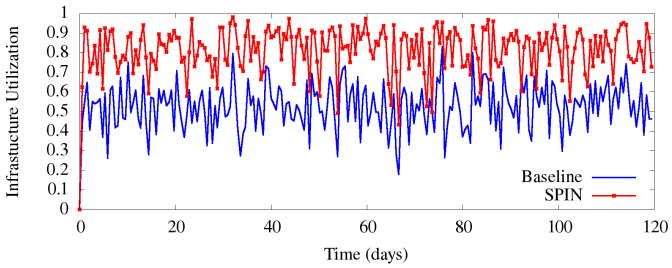

Fig. 12 – Infrastructure CPU utilization over time (request arrival rate: 0.03 rps)

- Infrastructure utilization: This is the amount of used CPU resource divided by the total available resource (CPU).
- Cumulative profit: This is computed as the revenue of the SFC provider minus its operational costs including the costs of the instance, bandwidth, and synchronization. The cumulative profit is computed for the duration of the experiment.
- Average end-to-end (e2e) delay: This is the average end-to-end delay between the sources and destinations of the SFC requests that were successfully embedded throughout the experiment.

In the following paragraphs, we provide and discuss the obtained results for each metric.
The first considered metric is the acceptance ratio and is depicted in Fig. 13. The figure shows that, even for a low request arrival rate, the baseline fails to accommodate 50% of the requests whereas SPIN succeeds in accommodating up to 65% of the requests. This means that even if the infrastructure's utilization is low and resources are available (Fig. 14), the baseline, unlike SPIN, is not able to efficiently leverage such available resources. As the arrival rate is increased, the acceptance ratio goes down for both algorithms as the infrastructure becomes saturated as Fig. 14. However, SPIN still outperforms the baseline in terms of acceptance ratio.
Furthermore, as illustrated in Fig. 14, SPIN accepts up to 25% more requests for low arrival rates, showing that it allows to efficiently leverage the infrastructure resources

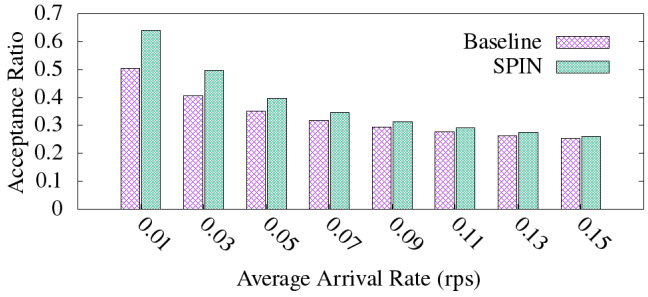

Fig. 13 – Acceptance ratio

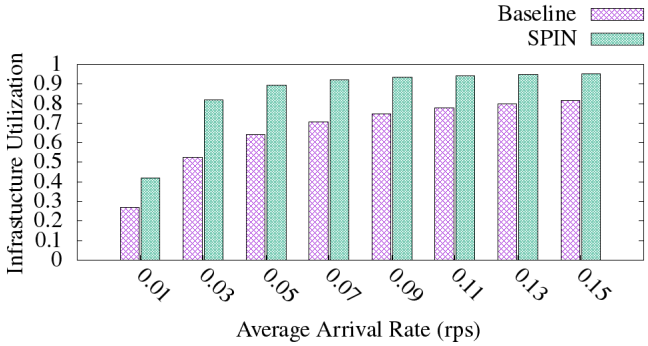

Fig. 14 – Infrastructure utilization

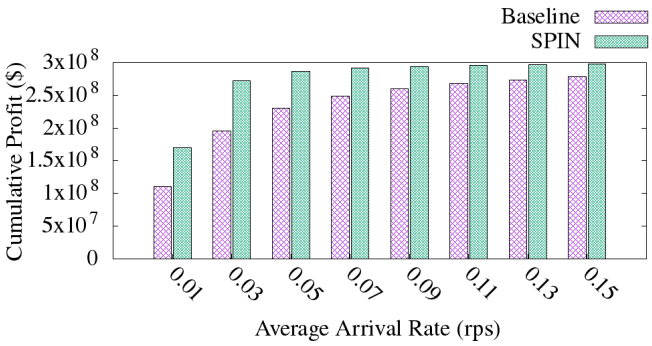

Fig. 15 – Cumulative profit

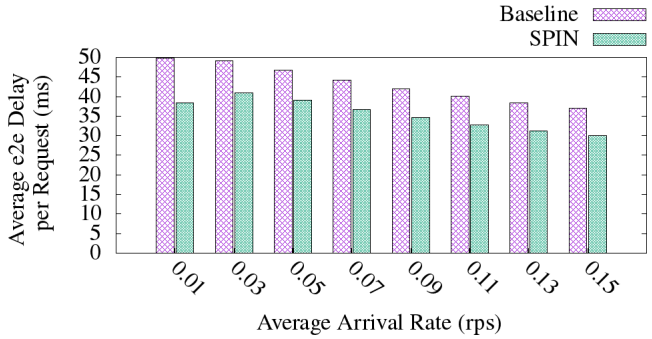

Fig. 16 – Average e2e delay per request

compared to the baseline. It is worth noting that for high arrival rates SPIN succeeds to reach 95% utilization compared to 80% utilization for the baseline. We also study cumulative profit generated by each of the two algorithms (Fig. 15). The figure shows that the profit generated by SPIN exceeds by up to 30% the one generated by the baseline, especially for low arrival rates. Finally, Fig. 16 shows the average end-to-end delay per request for the accepted SFCs. It shows that during low utilization, SPIN reduces by up to 35% e2e delay and by up to 25% the e2e delay for high arrival rates. This shows that SPIN does not only satisfy the requests' requirements in terms of e2e delay but further reduces it compared to the baseline.

8. CONCLUSION AND FUTURE DIRECTIONS

Selecting the right placement for the VNF, the number and the type of the VM instance is a major challenge for cloud providers as it has a paramount impact not only on performance but also on cost. In this paper, we started by studying these trade-offs using general-purpose Amazon EC2 instances. For instance, we found that micro-instances (small instances) are more cost-effective. We also find that the performance of a virtual machine is not always proportional to the amount of resources that are allo- cated. Hence, we found that, provisioning several small instances, when the function could be distributed, would provide better performance than big instances with a smaller cost.

Furthermore, we investigated pro it-driven resource allocation by mathematically modeling the problem as an integer linear program and proposing two heuristics, a baseline and a more sophisticated algorithm dubbed SPIN, which allows to improve the performance of the mapping with more accepted chains and hence to increase profits.

This work opens the door for more research opportunities. For instance, it would be interesting to further develop VNF benchmarks with the development of more sophisticated procedures to benchmark VNFs depending on the nature of the implemented network function. It is also of utmost importance to devise resource consump- tion models for specific VNFs taking into consideration the VNF characteristics and the hosting. Another research avenue is to assess synchronization costs among same- instance VNFs depending on the type of the function and the instance locations. More work should also be done on the management of VNFs by developing platform-aware resource allocation as the performance of a virtual machine significantly depends on the hosting platform (dedicated hardware versus software, server model, type and amount of resources).

REFERENCES

[1] Mohamed Faten Zhani and Hesham Elbakoury. "FlexNGIA: A Flexible Internet Architecture for the Next-Generation Tactile Internet". In: *Journal of Network and Systems Management, Springer* (2020).

[2] Alexander Clemm, Mohamed Faten Zhani, and Raouf Boutaba. "Network Management 2030: Operations and Control of Network 2030 Services". In: *Journal of Network and Systems Management, Springer* (2020).

[3] Blesson Varghese, Philipp Leitner, Suprio Ray, Kyle Chard, Adam Barker, Yehia Elkhatib, Herry Herry, Cheol-Ho Hong, Jeremy Singer, Fung Po Tso, Eiko Yoneki, and Mohamed Faten Zhani. "Cloud Futurology". In: *IEEE Computer* 52.9 (Sept. 2019), pp. 68–77.

[4] Walid Racheg, Nadir Ghrada, and Mohamed Faten Zhani. "Profit-driven resource provisioning in NFV-based environments". In: *IEEE International Conference on Communications (ICC)*. 2017, pp. 1–7.

[5] Farzad Tashtarian, Mohamed Faten Zhani, Bita Fatemipour, and Delaram Yazdani. "CoDeC: A Cost-Effective and Delay-Aware SFC Deployment". In: *IEEE Transactions on Network and Service Management* 17.2 (2020), pp. 793–806.

[6] Ahmed Amokrane, Mohamed Faten Zhani, Rami Langar, Raouf Boutaba, and Guy Pujolle. "Greenhead: Virtual data center embedding across distributed infrastructures". In: *IEEE transactions on cloud computing (TCC)* 1.1 (2013), pp. 36–49.

[7] *Amazon EC2 instances*. https://aws.amazon.com/fr/ec2/instance-types/. URL: https://aws.amazon.com/fr/ec2/instance-types/.

[8] Nadir Ghrada, Mohamed Faten Zhani, and Yehia Elkhatib. "Price and Performance of Cloud-hosted Virtual Network Functions: Analysis and Future Challenges". In: *IEEE Performance Issues in Virtualized Environments and Software Defined Networking (PVE-SDN NetSoft 2018)*. Montreal, Canada, June 2018.

[9] Zakaria Alomari, Mohamed Faten Zhani, Moayad Aloqaily, and Ouns Bouachir. "On Minimizing Synchronization Cost in NFV-based Environments". In: *IEEE/ACM/IFIP International Conference on Network and Service Management (CNSM)*. Virtual Conference, Nov. 2020.

[10] *AWS Infrastructure*. https://infrastructure.aws. Web Page. 2021. URL: https://infrastructure.aws.

[11] *Google Cloud*. https://cloud.google.com/. Web Page. 2021. URL: https://cloud.google.com/.

[12] Milad Ghaznavi, Aimal Khan, Nashid Shahriar, Khalid Alsubhi, Reaz Ahmed, and Raouf Boutaba. "Elastic virtual network function placement". In: *IEEE International Conference on Cloud Networking (CloudNet)*. 2015.

[13] Walid Racheg, Nadir Ghrada, and Mohamed Faten Zhani. "Profit-driven resource provisioning in NFV-based environments". In: *IEEE Conf. on Communications (ICC)*. 2017. DOI: 10.1109/ICC.2017.7997163.

[14] Francisco Carpio, Samia Dhahri, and Admela Jukan. "VNF placement with replication for Loac balancing in NFV networks". In: *IEEE International Conference on Communications (ICC)*. 2017, pp. 1–6.

[15] Xiaoke Wang, Chuan Wu, Franck Le, Alex Liu, Zongpeng Li, and Francis Lau. "Online VNF scaling in datacenters". In: *IEEE International Conference on Cloud Computing (CLOUD)*. 2016, pp. 140–147.

[16] Prajeesh Murukan, Dana Jamaluddine, Shalaka Kolhapure, Fady Mikhael, and Shiva Nouzari. "A Cost-based Placement Algorithm for Multiple Virtual Security Appliances in Cloud using SDN: MOUFLP (Multi-Ordered Uncapacitated Facility Location Problem)". In: *arXiv preprint arXiv:1602.08155* (2016).

[17] Xiaoxi Zhang, Chuan Wu, Zongpeng Li, and Francis CM Lau. "Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization". In: *IEEE Conference on Computer Communications (INFOCOM)*. 2017, pp. 1–9.

[18] Lav Gupta, Mohammed Samaka, Raj Jain, Aiman Erbad, Deval Bhamare, and Chris Metz. "COLAP: A predictive framework for service function chain placement in a multi-cloud environment". In: *IEEE Annual Computing and Communication Workshop and Conference (CCWC)*. 2017, pp. 1–9.

[19] Wenrui Ma, Jonathan Beltran, Zhenglin Pan, Deng Pan, and Niki Pissinou. "SDN-based traffic aware placement of NFV middleboxes". In: *IEEE Transactions on Network and Service Management* 14.3 (2017), pp. 528–542.

[20] Defang Li, Peilin Hong, Kaiping Xue, and Jianing Pei. "Availability aware VNF deployment in datacenter through shared redundancy and multi-tenancy". In: *IEEE Transactions on Network and Service Management* 16.4 (2019), pp. 1651–1664.

[21] Akanksha Patel, Mythili Vutukuru, and Dilip Krishnaswamy. "Mobility-aware VNF placement in the LTE EPC". In: *IEEE conference on network function virtualization and software defined networks (NFVSDN)*. 2017, pp. 1–7.

[22] Ke Yang, Hong Zhang, and Peilin Hong. "Energy-aware service function placement for service function chaining in data centers". In: *IEEE Global Communications Conference (GLOBECOM)*. 2016, pp. 1–6.

[23] Deval Bhamare, Mohammed Samaka, Aiman Erbad, Raj Jain, Lav Gupta, and H Anthony Chan. "Optimal virtual network function placement in multi-cloud service function chaining architecture". In: *Computer Communications* 102 (2017), pp. 1–16.

[24] Nicolas El Khoury, Sara Ayoubi, and Chadi Assi. "Energy-aware placement and scheduling of network traffic flows with deadlines on virtual network functions". In: *IEEE International Conference on Cloud Networking (Cloudnet)*. 2016, pp. 89–94.

[25] Dilip Krishnaswamy, Ravi Kothari, and Vijay Gabale. "Latency and policy aware hierarchical partitioning for NFV systems". In: *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. 2015, pp. 205–211.

[26] Siri Kim, Yunjung Han, and Sungyong Park. "An energy-aware service function chaining and reconfiguration algorithm in NFV". In: *IEEE International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. 2016, pp. 54–59.

[27] Mohammad M Tajiki, Stefano Salsano, Luca Chiaraviglio, Mohammad Shojafar, and Behzad Akbari. "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining". In: *IEEE Transactions on Network and Service Management* 16.1 (2018), pp. 374–388.

[28] Zhichao Xu, Xiaoning Zhang, Shui Yu, and Ji Zhang. "Energy-efficient virtual network function placement in telecom networks". In: *IEEE International Conference on Communications (ICC)*. 2018, pp. 1–7.

[29] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. "Virtual network functions placement and routing optimization". In: *IEEE International Conference on Cloud Networking (CloudNet)*. 2015, pp. 171–177.

[30] Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, Raouf Boutaba, and Otto Carlos Muniz Bandeira Duarte. "Orchestrating virtualized network functions". In: *IEEE Transactions on Network and Service Management* 13.4 (2016), pp. 725–739.

[31] Chuan Pham, Nguyen H Tran, Shaolei Ren, Walid Saad, and Choong Seon Hong. "Traffic-aware and energy-efficient VNF placement for service chaining: Joint sampling and matching approach". In: *IEEE Transactions on Services Computing* 13.1 (2017), pp. 172–185.

[32] Guto Leoni Santos, Diego de Freitas Bezerra, É lisson da Silva Rocha, Leylane Ferreira, André Luis Cavalcanti Moreira, Glauco Estácio Gonçalves, Maria Valéria Marquezini, Ákos Recse, Amardeep Mehta, Judith Kelner, et al. "Service Function Chain Placement in Distributed Scenarios: a Systematic Review". In: *Journal of Network and Systems Management* 30.1 (2022), pp. 1–39.

[33] Md. Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba. "nf.io: A file system abstraction for NFV orchestration". In: *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. 2015.

[34] Michael Till Beck and Juan Felipe Botero. "Coordinated Allocation of Service Function Chains". In: *IEEE Global Communications Conference (GLOBECOM)*. Dec. 2015, pp. 1–6.

[35] Mingshu Lu, Zhihui Wu, Da Li, and Dong Wang. "Resource Allocation Algorithm of Power Communication Network Service Function Chain based on Resource Characteristics". In: *IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*. 2021, pp. 382–387.

[36] Masahiro Sasabe and Takanori Hara. "Capacitated shortest path tour problem-based integer linear programming for service chaining and function placement in NFV networks". In: *IEEE Transactions on Network and Service Management* 18.1 (2020), pp. 104–117.

[37] Yun Wang, Chih-Kai Huang, Shan-Hsiang Shen, and Ge-Ming Chiu. "Adaptive Placement and Routing for Service Function Chains With Service Deadlines". In: *IEEE Transactions on Network and Service Management* 18.3 (2021), pp. 3021–3036.

[38] Chao Bu, Jinsong Wang, and Xingwei Wang. "Towards delay-optimized and resource-efficient network function dynamic deployment for VNF service chaining". In: *Elsevier Applied Soft Computing* 120 (2022), p. 108711.

[39] Tuan-Minh Pham. "Traffic Engineering Based on Reinforcement Learning for Service Function Chaining With Delay Guarantee". In: *IEEE Access* 9 (2021).

[40] *Amazon EC2 instances*. Upper Limits on Number of Amazon EC2 Instances by Region. Web Page. last accessed May 2021. URL: `https://alestic.com/2011/08/ec2-max-instances/`.

[41] *Shorewall Firewall*. `http://shorewall.org`. Accessed: 06-05-2021. 2021.

[42] *Snort - Network Intrusion and Prevention System*. `https://www.snort.org/`. Accessed: 06-05-2021. 2021.

AUTHORS

Tarik Moufakir received the Ph.D. degree in information technology in 2021 from École de Technologie Supérieure (ÉTS) in Montreal, QC, Canada. He received an M.Sc. in computer science in 2013 from UQAM University, Canada and another M.Sc. in computer science from ENST Bretagne, France in 2002. His main research interests are software-defined networking and cloud computing.

Mohamed Faten Zhani is an associate professor with the Software and IT Engineering Department at lÉcole de Technologie Supérieure (ÉTS), Montreal, QC, Canada. His research interests include cloud computing, network function virtualization, software-defined networking and resource management.

Abdelouahed Gherbi received the Ph.D. degree in computer engineering from Concordia University, Canada. He is currently an associate professor with the Software and IT Engineering Department, École de Technologie Supérieure (ÉTS), Montreal, QC, Canada. His research interests include model-driven software engineering, modeling and analysis techniques for real-time and critical software systems, software performance, high availability, and security.

Moayad Aloqaily is an assistant professor with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), United Arab Emirates. He received his Ph.D. degree in electrical and computer engineering from the University of Ottawa, Ontario, Canada. His current research interests include the applications of AI and ML, connected and autonomous vehicles, blockchain solutions, and sustainable energy and data management.

Nadir Ghrada received an M.Sc. in computer science in 2018 from École de Technologie Supérieure (ÉTS) in Montreal, QC, Canada. His main research interest is cloud computing.