Impact of Processing Costs on Service Chain Placement in Network Functions Virtualization

Marco Savi, Massimo Tornatore, Giacomo Verticale
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, Italy
firstname.lastname@polimi.it

***Abstract*—The Network Functions Virtualization (NFV) paradigm is the most promising technique to help network providers in the reduction of capital and energy costs. The deployment of virtual network functions (VNFs) running on generic x86 hardware allows higher flexibility than the classical middleboxes approach. NFV also reduces the complexity in the deployment of network services through the concept of service chaining, which defines how multiple VNFs can be chained together to provide a specific service. As a drawback, hosting multiple VNFs in the same hardware can lead to scalability issues, especially in the processing-resource sharing. In this paper, we evaluate the impact of two different types of costs that must be taken into account when multiple chained VNFs share the same processing resources: the upscaling costs and the context switching costs. Upscaling costs are incurred by VNFs multi-core implementations, since they suffer a penalty due to the needs of load balancing among cores. Context switching costs arise when multiple VNFs share the same CPU and thus require the loading/saving of their context. We model through an ILP problem the evaluation of such costs and we show their impact in a VNFs consolidation scenario, when the x86 hardware deployed in the network is minimized.**

I. Introduction

Current fixed and mobile network operators are struggling to reduce capital and energy costs and increase their revenues by smartly upgrading their network infrastructure. In the last years, an innovative network architecture paradigm called *Network Functions Virtualization* (NFV) [1] has emerged as a promising technique to help network operators in achieving this goal.

NFV is based on the concept of *network function*. A network function is an abstract building block whose aim is to process the network traffic to accomplish a specific task. Examples of network functions are Firewalls, Network Address Translators, Traffic Monitors etc. Nowadays, the network functions are usually implemented on dedicated hardware, usually referred as *middleboxes*. Such middleboxes are able to handle heavy traffic loads, but they are expensive and inflexible, since they are designed to perform limited operations on the traffic, and they must be dimensioned at peak loads, leading to waste of resources when the traffic is low, e.g., in off-peak hours.

The NFV paradigm consists in moving from an hardware to a software implementation of network functions in a virtualized environment. This way, multiple and heterogeneous *virtual network functions* (VNFs) can be hosted by the same generic x86 hardware. NFV adds flexibility to the network since it allows network operators to efficiently dimension and dynamically consolidate the VNFs. Another value added by NFV is the simplicity in the deployment of heterogeneous network services. In fact NFV can exploit the concept of *service chaining* [2], according to which a service (e.g., web browsing, VoIP, video streaming, online gaming etc.) can be provided by one or more service chains (SCs), i.e., a concatenation of appropriate VNFs that must be crossed by the traffic associated to that specific service. NFV has also some drawbacks. Moving from an hardware to a virtualized software implementation can lead to scalability issues in the resource sharing, especially concerning *processing*.

In this paper, we evaluate the impact of processing-resource sharing among multiple VNFs in a VNF consolidation scenario. We show how the *size* and the *number* of VNFs sharing the same x86 hardware influence the hardware processing performance, and how the deployment in the network of multiple SCs using such VNFs is affected. Indeed, consolidating multiple VNFs in the same hardware leads to inefficiencies due to the need of saving/loading the context (i.e., the state) of the VNFs sharing the processing resources (*context switching costs*) [3][4], as well as to inefficiencies due to the needs of VNF multi-core implementations (*upscaling costs*) [5]. To the best of our knowledge, this paper is the first study evaluating such processing-related costs in a VNF consolidation scenario.

The remainder of the paper is organized as follows. Section II discusses some related works. In Section III we introduce our system model, showing how we model the physical network, the VNFs, the SCs and the processing-related costs. Section IV introduces our optimization problem for the evaluation of the effects of upscaling and context switching costs in a VNF consolidation scenario. In Section V we show the numerical results and Section VI draws the conclusion of our work.

II. Related works

In our paper, we develop an optimization problem for SC and VNF placement that can be seen as an extension of some well-known Virtual Network Embedding (VNE) problems, as the ones shown in [6][7][8]. In our problem, the SCs can be seen as simple-chain virtual graphs, where the chained VNFs are virtual nodes connected together by virtual links. Such SCs must be embedded in a physical network, where each virtual link can be mapped to a physical path [6], multiple

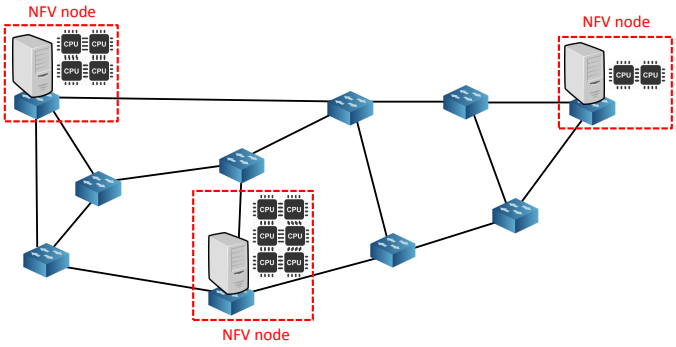

Fig. 1. Physical topology where some nodes are equipped with generic multi-core x86 hardware (i.e., *NFV nodes*)

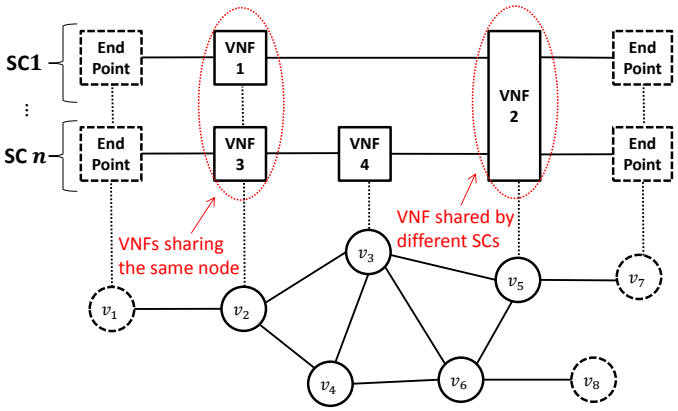

Fig. 2. Example of SCs that must be embedded in the physical network, where different VNFs can share the processing resources of the same NFV node and multiple SCs can share the same VNF

virtual nodes can be mapped to the same physical node [7], and the virtual nodes must be consolidated [8]. With respect to Refs. [6][7][8], in our SCs embedding problem it must be also guaranteed that a virtual node (i.e., a VNF) can be shared among multiple virtual networks (i.e., SCs).

Ref. [9] is the first work investigating the optimal placement of SCs in the network following a VNE approach. The authors first formalize the concept of SC and VNF, then they formulate a Mixed Integer Quadratically Constrained Problem to evaluate the optimal placement of VNFs and SCs. In our paper, we develop a similar optimization problem, but we extend the analysis to cover also the processing-related costs for the physical nodes.

Some other studies deal with efficient placing of SCs and VNFs in the network. Ref. [4] proposes an efficient algorithm for the communication among different chained VNFs that are hosted by the same generic hardware, while Ref. [10] introduces an algorithm to efficiently handle traffic load variations by dynamically instantiating VNFs. Ref. [11] defines an online algorithm for joint Virtual Machine placement, routing and consolidation with the aim of minimizing traffic costs. Our approach is similar, but we investigate the implications of such strategies from a processing-resource sharing perspective.

III. System Model

A. Physical topology and NFV nodes modeling

We model the physical network as a connected directed graph $\mathcal{G} = (V, E)$. All the network nodes $v \in V$ have forwarding capabilities. Then, the links $(v, v') \in E$ connecting the nodes are high-capacity fiber links. The network nodes v can also be equipped with standard x86 hardware (see Fig. 1), that can be used to host different VNFs. The x86 hardware can have different capabilities in terms of storage and processing, and can potentially be hosted in every powered physical location (e.g., in a cabinet, in a central office, in a core exchange, etc.). In this paper, we focus on the *processing* aspect and we generally call the nodes hosting generic x86 hardware *NFV nodes*. We assume that the NFV nodes are *multi-core* systems, and we measure the processing capability of each NFV node in terms of the number of CPU cores γ_v it is equipped with. Clearly, if $\gamma_v = 0$ the node v has not processing capability.

B. Virtual network functions modeling

In general, a VNF $f \in F$ can be seen as a black-box performing some operations on the network traffic. Every VNF f hosted by a NFV node v must be assigned a dedicated processing capability $c_{f,v}$ in order to fulfil all the operations it has to perform on the input traffic. The processing capability $c_{f,v}$ represents the fraction of the overall processing capability γ_v of the node v assigned to the VNF f, expressed in terms of number of CPU cores. We say that a VNF has a larger *size* when it is assigned a larger processing capability $c_{f,v}$. We assume that a VNF f can be shared among different SCs (e.g., VNF 2 in Fig. 2) each requiring a fraction π_f of the processing capability $c_{f,v}$. Note that heterogeneous VNFs require a π_f that can largely vary from one to each other, depending on the complexity of the operations performed. In our work we assume that π_f has been optimally chosen, i.e., for each SC, traffic is queued for a negligible time before it is processed. In other words, every VNF is designed in such a way it does not introduce additional latency due to a bad or inadequate implementation.

C. Service chains modeling

When a *service* is required between two end-points, one or more SCs $c \in C$ must be deployed in the network. In general, heterogeneous SCs can be deployed, and some SCs can be shared among different services. Without any loss of generality, we consider a one-to-one correspondence between a service and a SC. A SC c can be modelled by a simple-chain graph $\mathcal{C}^c = (X^c \cup U^c, G^c)$, where X^c is the set of start/end points u, U^c is the set of VNF requests u and G^c is the set of virtual links (u, u') chaining consecutive VNF requests/end points u and u'. From a topological point of view, both VNF requests and end points are virtual nodes $u \in X^c \cup U^c$. Note that, similarly to [9], in our model we decouple the concepts of *VNF* f and of *VNF request* u: for every SC c, every VNF request $u \in U^c$ is mapped to a specific VNF f. We then introduce the mapping parameter $\tau_u^c \in F$ to specify the mapping of the request u for the SC c to a specific VNF f. Similar considerations can be done for the end points $u \in X^c$.

We assume that such end points are fixed in the network, and we introduce the mapping parameter $\eta_u^c \in V$ to specify the mapping between the end point u for the SC c to a specific physical node v.

In our model, every single SC c can serve an aggregate set of *users*. Such aggregate SCs are deployed when multiple users require the same service between the same two end points. Every SC is also associated to some performance/QoS parameters such as:

- The *requested bandwidth* $\delta_{u,u'}^c$, i.e., the bandwidth that must be guaranteed between two VNF requests/end points u and u' to provide the service offered by the SC c. Note that every virtual link (u, u') can be associated to a different bandwidth requirement because the chained VNFs can lead to a change in the traffic throughput.
- The *maximum tolerated latency* φ^c, i.e., the maximum delay that can be introduced by the network without affecting the service between the two end points of the SC c.

D. Processing-related costs for the NFV nodes

The processing-related costs (upscaling and context switching) for a NFV node v are mainly influenced by:

- The *size* of the VNFs f hosted by the node v;
- The *number* of VNF requests u mapped to the VNFs f, which share the processing resources of the NFV node v.

Upscaling costs: When a VNF f must process a high quantity of traffic, it might need more processing resources than the ones provided by a single CPU core. In this case, a multi-core implementation of the VNF is needed and some upscaling costs must be taken into account. These costs arise because of the software architectural challenges that must be addressed when a multi-core VNF is implemented. In fact, the traffic that must be handled by multi-core VNFs needs to be balanced among different CPU cores by a *load balancer*. The needs of load balancing add a new layer in the system architecture that can become a bottleneck, must be carefully designed and leads to some performance penalties [12]. For this reason, the higher is the number of CPU cores used by the multi-core VNF, the higher are the upscaling costs. We then assume that such costs can be modelled as a step function of the number of CPU cores used by the VNF.

Context switching costs: When multiple VNFs are hosted by the same NFV node, the CPU cores are shared among such different VNFs. This leads to another kind of costs, i.e., the context switching costs. These costs are related to the effort that must be taken into account in order to execute multiple VNFs in the same CPU, and we assume that they linearly increase with the number of VNF requests mapped into the VNFs that share the CPU cores γ_v of the NFV node v. These costs are strictly related to the needs of cache sharing among different VNFs and of saving/loading the context (i.e., the state) of a VNF for CPU resource sharing [13][14].

These considerations show how a size/number trade-off on the VNFs sharing the same node exists, since a node hosting a big number of small VNFs must deal with big context switching costs and small upscaling costs, while a node hosting a small number of big VNFs leads to small context switching costs but big upscaling costs. We assume that the upscaling and context switching costs lead to two different performance degradation effects:

- Increase of the *latency* introduced by the NFV node. In fact, both deciding how to balance traffic among different CPU cores and saving/loading the VNFs context require some computational time. We call μ_v the upscaling latency and ω_v the context switching latency for the NFV node v.
- Decrease of the *actual processing capability* of the NFV node. In fact, both taking a decision about how to balance traffic among different cores and saving/loading the VNFs context require some dedicated processing capability, that in turn cannot be used by the VNFs hosted in that node. We call κ_v the upscaling processing and ξ_v the context switching processing for the NFV node v.

TABLE I
SUMMARY OF GRAPHS AND SETS CONSIDERED IN THE MODEL

Graph/Set	Description
$\mathcal{G} = (V, E)$	Physical network graph, where V is the set of physical nodes v and E is the set of physical links (v, v') connecting the nodes v and v'
C	Set of the service chains c that must be embedded in the physical network $\mathcal{G}$
$\mathcal{C}^c = (X^c \cup U^c, G^c)$	Simple-chain graph for the SC c, where X^c is the set of fixed start/end point u, U^c is the set of VNF requests u, G^c is the set of virtual links (u, u') connecting the VNF request (or start point) u and the VNF request (or end point) u'
F	Set of VNFs f that can be requested and deployed in the network

TABLE II
SUMMARY OF PARAMETERS CONSIDERED IN THE MODEL

Parameter	Domain	Description
τ_u^c	$c \in C$ $u \in U^c$	VNF requested by the VNF request u in the SC c ($\tau_u^c \in F$)
η_u^c	$c \in C$ $u \in X^c$	Physical node where the start/end point u for the SC c is mapped to ($\eta_u^c \in V$)
π_f	$f \in F$	Fraction of the CPU processing required by each VNF request u for the VNF f
γ_v	$v \in V$	Number of the CPU cores hosted by the node v
$\beta_{v,v'}$	$(v, v') \in E$	Bandwidth capacity of the physical link (v, v')
$\lambda_{v,v'}$	$(v, v') \in E$	Latency of the physical link (v, v')
μ_v	$v \in V$	Upscaling latency of the node v
ω_v	$v \in V$	Context switching latency of the node v
κ_v	$v \in V$	Upscaling processing of the node v
ξ_v	$v \in V$	Context switching processing of the node v
$\delta_{u,u'}^c$	$c \in C$ $(u, u') \in G^c$	Requested bandwidth on the virtual link (u, u') for the SC c
φ^c	$c \in C$	Maximum tolerated latency by the SC c
$\mathcal{M}$		Big-M parameter

TABLE III
DECISION VARIABLES FOR THE ILP MODEL

Variable	Domain	Description
$m^c_{u,v} \in \{0,1\}$	$c \in C$ $u \in U^c$ $v \in V$	Binary variable such that $m^c_{u,v} = 1$ iff the VNF request u for the SC c is mapped to the node v, otherwise $m^c_{u,v} = 0$
$c_{f,v} \in [0, \gamma_v]$	$f \in F$ $v \in V$	Real variable indicating the fraction of the CPU cores in the node v used by the VNF f
$i_{f,v} \in \{0,1\}$	$f \in F$ $v \in V$	Binary variable such that $i_{f,v} = 1$ iff the VNF f is hosted by the node v, otherwise $i_{f,v} = 0$
$e^c_{v,v',x,y,u,u'} \in \{0,1\}$	$c \in C$ $(v,v') \in E$ $x \in V$ $y \in V$ $(u,u') \in G^c$	Binary variable such that $e^c_{v,v',x,y,u,u'} = 1$ iff the physical link (v,v') belongs to the path between the nodes x and y, where the VNF requests u and u' for the SC c are mapped to, otherwise $e^c_{v,v',x,y,u,u'} = 0$
$a_v \in \{0,1\}$	$v \in V$	Binary variable such that $a_v = 1$ iff the node v is active, otherwise $a_v = 0$

IV. OPTIMIZATION PROBLEM FOR VNFS CONSOLIDATION

In this Section we formulate an Integer Linear Programming (ILP) model for VNF consolidation. A summary of the sets, parameter and variables used in the model is reported in Tables I, II, III. Given a physical network topology and some SCs, we want to optimally decide the position and the size of the chained VNFs while minimizing the number of *active* NFV nodes (i.e., the nodes hosting at least one VNF) in the network. The constraints are grouped in three categories: *requests placement* constraints, *routing* constraints and *performance* constraints. The requests placement constraints ensure a correct mapping of the VNFs f on the NFV nodes v as well as a correct mapping between the VNF requests u and the VNFs f. The routing constraints guarantee a correct mapping of the virtual links (u,u') on physical paths constituted by different links (v,v'). Finally, the performance constraints are related to all the performance requirements that must be addressed in the network.

1) Objective function:

$$\min \sum_{v \in V} a_v \tag{1}$$

The objective function minimizes the number of active NFV nodes. This way, we try to consolidate as much as possible the VNFs. This optimization problem can be useful for network operators to plan the best strategical placement of the x86 hardware. The number of active nodes in the network is a measure of the cost for NFV implementation.

2) Requests placement constraints:

$$m^c_{u,\eta^c_u} = 1 \quad c \in C, u \in X^c \tag{2}$$

$$m^c_{u,v} = 0 \quad c \in C, u \in X^c, v \in V : v \neq \eta^c_u \tag{3}$$

The fixed start/end point u of a SC c is mapped to the node v specified by the parameters η^c_u (constraint 2) and it is not mapped to any other node (constraint 3).

$$\sum_{v \in V} m^c_{u,v} = 1 \quad c \in C, u \in U^c \tag{4}$$

Every VNF request u for each SC c must be mapped to exactly one node v.

$$\sum_{\substack{c \in C \\ u \in U^c : \tau^c_u = f}} m^c_{u,v} \leq \frac{c_{f,v}}{\pi_f} \quad f \in F, v \in V \tag{5}$$

The overall number of VNF requests u mapped to the VNF f hosted by the node v cannot overcome $c_{f,v}/\pi_f$, i.e., the maximum number of VNF requests u for the VNF f that can be mapped to that node, according to the processing requirement π_f per request u and the fraction of CPU cores $c_{f,v}$ of the node v assigned to the VNF f.

$$c_{f,v} \leq \mathcal{M} \cdot i_{f,v} \quad f \in F, v \in V \tag{6}$$

$$i_{f,v} - c_{f,v} < 1 \quad f \in F, v \in V \tag{7}$$

Constraints 6 and 7 ensure that $i_{f,v} = 0$ if $c_{f,v} = 0$ and that $i_{f,v} = 1$ if $c_{f,v} > 0$. $i_{f,v}$ is a flag variable used to understand if the VNF f is mapped to the node v. $\mathcal{M}$ is a big-M parameter, greater than the maximum value taken by $c_{f,v}$, i.e., $\mathcal{M} > \max_{v \in V}\{\gamma_v\}$.

$$i_{f,v} \leq \sum_{\substack{c \in C \\ u \in U^c : \tau^c_u = f}} m^c_{u,v} \quad f \in F, v \in V \tag{8}$$

The VNF f is placed on a node v only if there is at least one request u mapped to it.

$$\sum_{f \in F} c_{f,v} \leq \gamma_v - \psi_v \quad v \in V \tag{9}$$

For each node v, the overall CPU processing assigned to the VNFs f cannot overcome the actual processing capability of the node, expressed in terms of CPU cores. The actual CPU processing capability of the node v is the difference between the CPU processing capacity γ_v of the node and the overall upscaling/context switching processing costs ψ_v. Such costs are taken into account every time a VNF request u is mapped to a VNF f. We define ψ_v in the following way:

$$\psi_v = \sum_{\substack{c \in C, \ f \in F \\ u \in U^c : \tau^c_u = f}} m^c_{u,v} \cdot (\lceil c_{f,v} \rceil \cdot \kappa_v + \sum_{g \in F} i_{g,v} \cdot \xi_v) \quad v \in V \tag{10}$$

The right-hand side of ψ_v requires the product between the binary variable $m^c_{u,v}$ and the real variable $\lceil c_{f,v} \rceil \cdot \kappa_v + \sum_{g \in F} i_{g,v} \cdot \xi_v$. Such product can be linearized, as well as the ceiling function $\lceil c_{f,v} \rceil$.

3) Routing constraints:

$$e^c_{v,v',x,y,u,u'} \leq m^c_{u,x} \cdot m^c_{u',y}$$
$$c \in C, (v,v') \in E, x \in V, y \in V, (u,u') \in G^c \tag{11}$$

A physical link (v,v') can belong to a path between two nodes x and y for a virtual link (u,u') of the SC c only if the two consecutive VNF requests or start/end points u and u' are mapped to these nodes. The product $m^c_{u,x} \cdot m^c_{u',y}$ of binary

variables can be linearized.

$$\sum_{(x,v)\in E,\ x,y\in V} e^c_{x,v,x,y,u,u'} \cdot m^c_{u,x} \cdot m^c_{u',y} = 1 \quad c \in C, (u,u') \in G^c \tag{12}$$

$$\sum_{(v,y)\in E,\ x,y\in V} e^c_{v,y,x,y,u,u'} \cdot m^c_{u,x} \cdot m^c_{u',y} = 1 \quad c \in C, (u,u') \in G^c \tag{13}$$

The virtual link (u, u') between two consecutive VNF requests or start/end points u and u' starts in one of the links connected to the node x, where the VNF request or start point u is mapped to (constraint 12), and it ends in one of the links of the node y, where the VNF request or end point u' is mapped to (constraint 13). These constraints are called *source* and *destination* constraints, and the products $e^c_{x,v,x,y,u,u'} \cdot m^c_{u,x} \cdot m^c_{u',y}$ and $e^c_{v,y,x,y,u,u'} \cdot m^c_{u,x} \cdot m^c_{u',y}$ of binary variables can be linearized.

$$\sum_{(v,x)\in E,\ v\in V} e^c_{v,x,x,y,u,u'} = 0$$
$$c \in C, x \in V, y \in V, x \neq y, (u,u') \in G^c \tag{14}$$

$$\sum_{(y,v)\in E,\ v\in V} e^c_{y,v,x,y,u,u'} = 0$$
$$c \in C, x \in V, y \in V, x \neq y, (u,u') \in G^c \tag{15}$$

While mapping the virtual link (u, u') for the SC c on a physical path between the nodes x and y where the VNF requests or start/end points u and u' are mapped to, no incoming link for the node x (constraint 14) and no outgoing link for the node y (constraint 15) is considered.

$$\sum_{(v,w)\in E,\ v\in V} e^c_{v,w,x,y,u,u'} = \sum_{(w,v')\in E,\ v'\in V} e^c_{w,v',x,y,u,u'}$$
$$c \in C, w \in V, x \in V, y \in V, x \neq w, y \neq w, (u,u') \in G^c \tag{16}$$

$$\sum_{(v,w)\in E,\ v\in V} e^c_{v,w,x,y,u,u'} \leq 1$$
$$c \in C, w \in V, x \in V, y \in V, x \neq w, y \neq w, (u,u') \in G^c \tag{17}$$

While considering a generic node w (other than the source node x and the destination node y of a virtual link (u, u')), if one of its incoming links belongs to the path between the nodes x and y, then also one of its outgoing links must belong to the path (constraint 16). Without constraint 17 multiple incoming/outgoing links could be considered, but we deal with unsplittable flows. Constraints 16 and 17 are called *transit* constraints.

$$e^c_{v,v,x,y,u,u'} = 0$$
$$c \in C, v \in V, x \in V, y \in V, x \neq y, (u,u') \in G^c \tag{18}$$

$$e^c_{v,v',x,x,u,u'} = 0$$
$$c \in C, x \in V, (u,u') \in G^c, (v,v') \in E, v \neq v' \tag{19}$$

In the physical network $\mathcal{G} = (V, E)$, every node v is connected to a self-loop (v, v) with infinite bandwidth ($\beta_{v,v} = \infty$) and zero latency ($\lambda_{v,v} = 0$). This self-loop is used when two consecutive VNF requests or start/end points u and u' are mapped to the same node x, and cannot be used otherwise (constraint 18). Moreover, no physical link (v, v') different from the self-loop is used when the VNF requests or start/end points u and u' are mapped to the same node x (constraint 19).

TABLE IV
REQUIREMENTS FOR THE DEPLOYED SERVICE CHAINS

Service Chain	Chained VNFs	δ	φ
Web Service	NAT-FW-TM-WOC-IDPS	100 kbit/s	500 ms
VoIP	NAT-FW-TM-FW-NAT	64 kbit/s	100 ms
Video Streaming	NAT-FW-TM-VOC-IDPS	4 Mbit/s	100 ms
Online Gaming	NAT-FW-VOC-WOC-IDPS	50 kbit/s	60 ms

NAT: *Network Address Translator*, FW: *Firewall*, TM: *Traffic Monitor*, WOC: *WAN Optimization Controller*, IDPS: *Intrusion Detection Prevention System*, VOC: *Video Optimization Controller*

4) Performance constraints:

$$\sum_{\substack{c\in C,\ x,y\in V \\ (u,u')\in G^c}} e^c_{v,v',x,y,u,u'} \cdot \delta^c_{u,u'} \leq \beta_{v,v'} \quad (v,v') \in E \tag{20}$$

The overall bandwidth $\delta^c_{u,u'}$ requested by the virtual links (u, u') of every SC c and mapped to the physical link (v, v') cannot exceed the capacity of the link $\beta_{v,v'}$.

$$\sum_{\substack{(v,v')\in E,\ x,y\in V \\ (u,u')\in G^c}} e^c_{v,v',x,y,u,u'} \cdot l_{v,v'} + \sigma^c \leq \varphi^c \quad c \in C \tag{21}$$

The latency introduced by the network between the start and end point of a SC c cannot overcome the maximum tolerated latency φ^c. The first term of the left-hand side of constraint 21 is related to the latency introduced by all the physical links (v, v') of the paths to which the virtual links (u, u') are mapped, while the term σ^c is related to the latency introduced by the nodes because of upscaling and context switching. We define σ^c in the following way:

$$\sigma^c = \sum_{\substack{f\in F,\ v\in V \\ u\in U^c:\tau^c_u=f}} m^c_{u,v} \cdot (\lceil c_{f,v} \rceil \cdot \mu_v + \sum_{g\in F} i_{g,v} \cdot \omega_v) \quad c \in C \tag{22}$$

The right-hand side of σ^c requires the product between the binary variable $m^c_{u,v}$ and the real variable $\lceil c_{f,v} \rceil \cdot \mu_v + \sum_{g\in F} i_{g,v} \cdot \omega_v$. Such product can be linearized, as well as the ceiling function $\lceil c_{f,v} \rceil$.

$$\sum_{f\in F} i_{f,v} \leq \mathcal{M} \cdot a_v \quad v \in V \tag{23}$$

$$a_v \leq \sum_{f\in F} i_{f,v} \quad v \in V \tag{24}$$

Constraints 23 and 24 assure that a node is marked as *active* (i.e., $a_v = 1$) only if at least one VNF f is hosted by that node. The big-M parameter $\mathcal{M}$ must be chosen such that $\mathcal{M} > |F|$.

V. SIMULATION RESULTS

A. Description of the simulation settings

We solved the optimization problem shown is Section IV using CPLEX. We consider the physical network topology shown in Fig. 1 with ten physical nodes ($|V| = 10$). This network topology is taken form the Internet2 network [15], considering only the nodes with advanced layer 3 services. We defined four different types of SCs that can be deployed in the network (see Table IV). Such SCs represent four different services, i.e., Web Service, VoIP, Video Streaming and Online Gaming. For every SC the traffic flows through a given

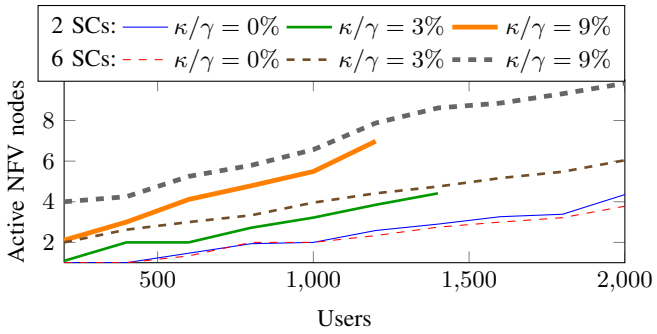

Fig. 3. Number of active NFV nodes as a function of the number of users in the network, while considering the impact of different *upscaling costs* and different numbers of deployed SCs in the *heterogeneous scenario*

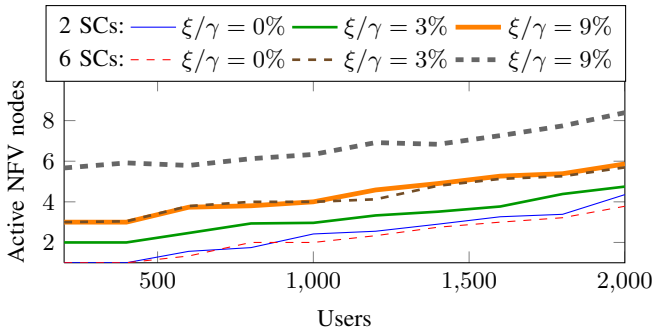

Fig. 4. Number of active NFV nodes as a function of the number of users in the network, while considering the impact of different *context switching costs* and different numbers of deployed SCs in the *heterogeneous scenario*

sequence of VNFs and two end points; some VNFs are shared among the different SCs. Table IV shows also the requirements in terms of bandwidth δ and maximum tolerated latency φ for each SC: we assume that every virtual link (u, u') of a SC requires the same bandwidth and that all the nodes can be both NFV nodes and start/end points for the SCs. Every SC can serve an aggregate traffic from multiple users. We assume that all the nodes can potentially host the same number γ of CPU cores and that they incur in the same upscaling costs κ and the same context switching costs ξ. We show our results considering two different scenarios: an *heterogeneous* scenario and an *homogeneous* scenario. In the heterogeneous scenario, we uniformly randomize the choice of start/end points (among the 10 nodes of the network) and of the deployed SCs (among the four different types of SCs). In the homogeneous scenario, we only uniformly randomize the choice of start/end points, while we assume that only one type of SC is deployed in the network.

B. Heterogeneous scenario

Figure 3 shows the impact of the *upscaling costs* κ (normalized to the overall processing capability γ of the node) as a function of the number of users and SCs deployed in the network when the context switching costs ξ are negligible. The total number of users in the network is equally split among the number of deployed SCs. We consider, for each point in the graph, 50 randomized instances and the deployment of 2

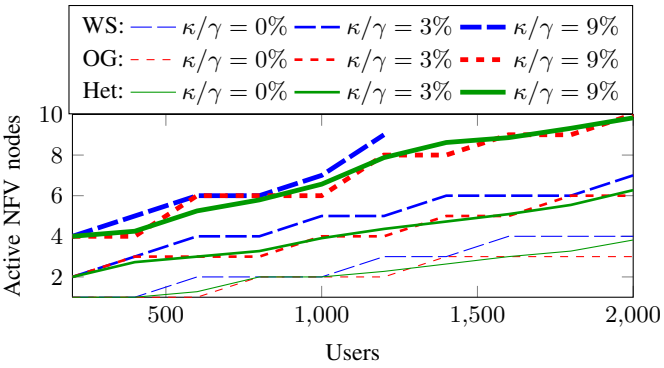

Fig. 5. Number of active NFV nodes as a function of the number of users in the network, while considering the impact of different *upscaling costs* in the *homogeneous scenario*, i.e., when only Web Service (WS) chains or Online Gaming (OG) chains are deployed (6 SCs in the network)

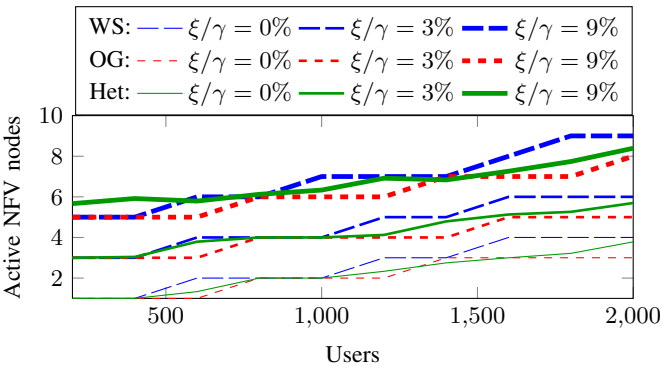

Fig. 6. Number of active NFV nodes as a function of the number of users in the network, while considering the impact of different *context switching costs* in the *homogeneous scenario*, i.e., when only Web Service (WS) chains or Online Gaming (OG) chains are deployed (6 SCs in the network)

SCs and 6 SCs. Each instance can be solved in few minutes. All the curves show a monotonically increasing trend. In fact, increasing the number of users in the network requires an additional processing capability and thus a higher number of active NFV nodes. If also the upscaling costs are negligible ($\kappa/\gamma = 0\%$), the number of NFV nodes in the network is almost the same as the number of SCs in the network. This means that if no processing costs (upscaling and context switching) are considered, increasing the number of deployed SCs does not affect the number of active NFV nodes in the network, as long as the overall number of users remains the same. When the upscaling costs are not negligible, increasing the number of SCs in the network has always an impact on the number of active NFV nodes ($\kappa/\gamma = 3\%$, $\kappa/\gamma = 9\%$), but the relative difference, considering the deployment of 2 SCs and 6 SCs, is almost the same for both $\kappa/\gamma = 3\%$ and $\kappa/\gamma = 9\%$ (around one active node). This happens because splitting the number of users among a higher number of SCs does not strongly impact on the size of the hosted VNFs. Note that with upscaling costs $\kappa/\gamma = 9\%$ some loads result in infeasible problems and, thus, no points are reported in the graph.

Figure 4 shows instead the impact of the *context switching costs* ξ when the upscaling costs are negligible. Also in this

case, the curves have a monotonically increasing trend. We can see then how an increase in the number of SCs deployed in the network has an impact not only in terms of the absolute number of active nodes in the network, but also in terms of the relative difference between the number of active nodes in the case of deployment of 2 SCs or 6 SCs. In fact, Fig. 4 shows how the difference in the number of active nodes is much higher for $\xi/\gamma = 3\%$ than for $\xi/\gamma = 9\%$: for $\xi/\gamma = 3\%$ such difference is around one active node, while for $\xi/\gamma = 9\%$ this difference is around three active nodes. This happens because the number of the overall VNFs requests is higher when 6 SCs are deployed than when only 2 SCs are deployed, and this leads to higher context switching costs.

C. Homogeneous scenario

For the homogeneous case, we consider two different cases: in the first case, only Web Service (WS) chains are deployed, in the second only Online Gaming (OG) chains. Figure 5 shows the impact of the *upscaling costs* in such cases, in comparison with the heterogeneous (Het) scenario. The number of deployed SCs is fixed at 6 SCs. The deployment of an homogeneous type of SC does not impact so much on the number of active nodes, in fact the curves of the homogeneous (WS and OG) and heterogeneous (Het) cases are more or less overlapped while considering the same value of κ/γ. In average, the deployment of OG chains impacts a bit less than the deployment of WS chains, for all the considered values of κ/γ, mainly due to the looser bandwidth requirements. While comparing such homogeneous cases with the heterogeneous case, we notice that the heterogeneous case behaves similarly to an average of the different homogeneous cases. Similar considerations can also be made while considering the *context switching costs* (Fig. 6).

Note that for both the heterogeneous and the homogeneous scenarios we simulated also the deployment of 3, 4 and 5 SCs. We do not plot such curves, since the results we got are analogous and do not provide additional insights to the discussion.

VI. Conclusion

In this paper, we investigated the impact of processing-resource sharing among VNFs and scalability costs in a NFV scenario, when multiple SCs must be deployed in the network. The VNFs placement and distribution on NFV nodes lead to two different types of costs: upscaling costs and context switching costs. Such costs lead to a VNF size/number trade-off that must be investigated. We first focused on the mathematical modeling of the NFV nodes, of the VNFs and of the SCs. Then, we defined an ILP model aimed at the VNF/SC embedding on a physical network while consolidating the VNFs in the minimum number of NFV nodes and taking into account VNF placement, routing and performance constraints. Then, we evaluated the impact of upscaling and context switching costs on the cost for NFV implementation. Results show that, as the number of SCs grows, the impact of context switching costs on the cost for NFV implementation is amplified. On the other hand, the number of SCs does not change how the upscaling costs translate into the cost for NFV implementation. We also showed how different latency and bandwidth requirements of the SCs impact on the network cost. Our model shows that NFV can handle multiple SCs with different requirements deployed on the same network without incurring in significant additional costs with respect to a scenario with homogeneous requirements. Several issues still remain open for future research. For example, more detailed models for upscaling and context switching costs can be investigated, including energetic aspects.

Acknowledgment

The research leading to these results has received funding from the European Community Seventh Framework Programme FP7/2013-2015 under grant agreement no. 317762 COMBO project.

References

[1] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng *et al.*, "Network Functions Virtualisation - Introductory White Paper," in *SDN and OpenFlow World Congress*, Oct. 2012.

[2] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in Network Service Chaining," in *IEEE Conference on SDN for Future Networks and Services*, Nov. 2013.

[3] I. Cerrato, G. Marchetto, F. Risso, R. Sisto, and M. Virgilio, "An efficient data exchange algorithm for chained network functions," in *IEEE International Conference on High Performance Switching and Routing*, Jul. 2014.

[4] I. Cerrato, M. Annarumma, and F. Risso, "Supporting fine-grained network functions through Intel DPDK," in *European Workshop on Software Defined Networks*, Sep. 2014.

[5] J. Wang, H. Cheng, B. Hua, and X. Tang, "Practice of parallelizing network applications on multi-core architectures," in *ACM International Conference on Supercomputing*, Jun. 2009.

[6] J. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, "Energy efficient virtual network embedding," *IEEE Communications Letters*, vol. 16, no. 5, pp. 756–759, May 2012.

[7] C. Fuerst, S. Schmid, and A. Feldmann, "Virtual network embedding with collocation: Benefits and limitations of pre-clustering," in *IEEE International Conference on Cloud Networking*, Nov. 2013.

[8] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE INFOCOM*, Apr. 2009.

[9] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE International Conference on Cloud Networking*, Oct. 2014.

[10] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *International Conference on Network and Service Management*, Nov. 2014.

[11] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *IEEE INFOCOM*, Mar. 2012.

[12] X. Geng, G. Xu, and Y. Zhang, "Dynamic load balancing scheduling model based on multi-core processor," in *International Conference on Frontier of Computer Science and Technology*, Aug. 2010.

[13] C. Li, C. Ding, and K. Shen, "Quantifying the cost of context switch," in *ACM Workshop on Experimental Computer Science*, Jun. 2007.

[14] R. Fromm and N. Treuhaft, "Revisiting the cache interference costs of context switching," 1996.

[15] "Internet2 network infrastructure topology," Oct. 2014. [Online]. Available: http://www.internet2.edu/media_files/422