Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2016, Article ID 7139852, 12 pages
http://dx.doi.org/10.1155/2016/7139852

Research Article

Server Resource Dimensioning and Routing of Service Function Chain in NFV Network Architectures

V. Eramo,[1] A. Tosti,[2] and E. Miucci[1]

[1]*DIET, "Sapienza" University of Rome, Via Eudossiana 18, 00184 Rome, Italy*
[2]*Telecom Italia, Via di Val Cannuta 250, 00166 Roma, Italy*

Correspondence should be addressed to E. Miucci; 1kronos1@gmail.com

Received 29 September 2015; Accepted 18 February 2016

Academic Editor: Vinod Sharma

The Network Function Virtualization (NFV) technology aims at virtualizing the network service with the execution of the single service components in Virtual Machines activated on Commercial-off-the-shelf (COTS) servers. Any service is represented by the Service Function Chain (SFC) that is a set of VNFs to be executed according to a given order. The running of VNFs needs the instantiation of VNF instances (VNFI) that in general are software components executed on Virtual Machines. In this paper we cope with the routing and resource dimensioning problem in NFV architectures. We formulate the optimization problem and due to its NP-hard complexity, heuristics are proposed for both cases of offline and online traffic demand. We show how the heuristics works correctly by guaranteeing a uniform occupancy of the server processing capacity and the network link bandwidth. A consolidation algorithm for the power consumption minimization is also proposed. The application of the consolidation algorithm allows for a high power consumption saving that however is to be paid with an increase in SFC blocking probability.

1. Introduction

Today's networks are overly complex, partly due to an increasing variety of proprietary, fixed-function appliances that are unable to deliver the agility and economics needed to address constantly changing market requirements [1]. This is because network elements have traditionally been optimized for high packet throughput at the expense of flexibility, thus hampering the deployment of new services [2]. Network Function Virtualization (NFV) can provide the infrastructure flexibility and agility needed to successfully compete in today's evolving communications landscape [3]. NFV implements network functions in software running on a pool of shared commodity servers instead of using dedicated proprietary hardware. This virtualized approach decouples the network hardware from the network function and results in increased infrastructure flexibility and reduced hardware costs. Because the infrastructure is simplified and streamlined, new and expended services can be created quickly and with less expense. Implementation of the paradigm has also been proposed [4] and the performance has been investigated [5]. To support the NVF technology both ETSI [6, 7] and IETF [8, 9] are defining novel network architectures able to allocate resources for Virtualized Network Function (VNF) as well as manage and orchestrate NFV to support services. In particular the service is represented by a Service Function Chain (SFC) [8] that is a set of VNFs that have to be executed according to a given order. Any VNF is run on a VNF instance (VNFI) implemented with one Virtual Machine (VM) whose resources (Vcores, RAM memory, etc.) are allocated to [10]. Some solutions have been proposed in the literature to solve the problem of choosing the servers where to instantiate VNF and to determine the network paths interconnecting the VNFs [1]. A formulation of the optimization problem is illustrated in [11]. Three greedy algorithms and a tabu search-based heuristic are proposed in [12]. The extension of the problem in the case in which virtual routers are also considered is proposed in [13].

The innovative contribution of our paper is that differently from [12] we follow an approach that allows for both the resource dimensioning and the SFC routing. We formulate the optimization problem whose objective is the minimization of the number of dropped SFC requests with the constraint that the SFCs are routed so as to respect both the server processing capacity and network link bandwidth. With the problem being NP-hard we introduce a heuristic that allows for (i) the dimensioning of the Virtual Machines in terms of number of Vcores assigned and (ii) the SFC routing through the VNF instance implemented by the Virtual Machines. The heuristic performs simultaneously the dimensioning and routing operations. Furthermore we propose a consolidation algorithm based on Virtual Machine migrations and able to achieve power consumption savings. The proposed algorithms are evaluated in scenarios characterized by offline and online traffic demands.

The paper is organized as follows. The related work is discussed in Section 2. Section 3 is devoted to illustrating the optimization problem and the proposed heuristic for the offline traffic demand case. In Section 4 we describe an SFC planner in which the proposed heuristic is applied in the case of online traffic demand. The planner also implements a consolidation algorithm whose application allows for power consumption savings. Some numerical results are shown in Section 5 to prove the effectiveness of the proposed heuristics. Finally the main conclusions and future research items are mentioned in Section 6.

2. Related Work

The Internet Engineering Task Force (IETF) has formed the SFC Working Group [8, 9] to define Service Function Chaining related problems and to standardize the architecture and protocols. A Service Function Chain (SFC) is defined as a set of abstract service functions [15] and ordering constraints that must be applied to packets selected as a result of the classification. When virtual service functions are considered, the SFC is referred to as Virtual Network Function Forwarding Graph (VNFFG) within the ETSI [6]. To support the SFCs, Virtual Network Function instances (VNFIs) are activated and executed in COTS servers. To achieve the economics of scale expected from NFV, network link and server resources should be used efficiently. For this reason efficient algorithms have to be introduced to determine where to instantiate the VNFI and to route the SFCs by choosing the network paths and the VNFI involved. The algorithms have to take into account the limited resources of the network links and the servers and pursued objectives of load balancing, energy saving, recovery from failure, and so on [1]. The task of placing SFC is closely related to virtual network embeddings [16] and virtual data network embedding [17] and may therefore be formulated as an optimization problem, with a particular objective. The approach has been followed by [11, 18–21]. For instance, Moens and Turck [11] formulate the SFC placement problem as a linear optimization problem that has the objective of minimizing the number of active servers. Other objectives are pursued in [19] (latency, remaining data rate, number of used network nodes, etc.) and the SFC placing is formulated as a mixed integer quadratically constrained program.

It has been proved that the SFC placing problem is NP-hard. For this reason efficient heuristics have been proposed and evaluated [10, 13, 22, 23]. For example, Xia et al. [22] formulate the placement and chaining problem as binary integer programming and propose a greedy heuristic in which the SFCs are first sorted according to their resource demands and the SFCs with the highest resource demands are given priority for placement and routing.

In order to achieve energy consumption saving, the NFV architecture should allow for migrations of VNFI, that is, the migration of the Virtual Machine implementing the VNFI. Though VNFI migrations allow for energy consumption saving, they may impact the QoS performance received by the users related to the migrated VNFs. A model has been proposed in [24] to derive some performance indicators, such as the whole service down time and the total migration time so as to make function migrations decisions.

The contribution of this paper is twofold: (i) to propose and to evaluate the performance of an algorithm that performs simultaneously resource dimensioning and SFC routing and (ii) to investigate the advantages from the point of view of the power consumption saving that the application of server consolidation techniques allow us to achieve.

3. Offline Algorithms for SFC Routing in NFV Network Architectures

We consider the case in which SFC requests are known in advance. We formulate the optimization problem whose objective is the minimization of the number of dropped SFC requests with the constraint that the SFCs are routed so as to respect both the server processing capacity and network link bandwidth. With the problem being NP-hard we introduce a heuristic that allows for (i) the dimensioning of the Virtual Machines in terms of the number of Vcores assigned and (ii) the SFC routing through the VNF instances implemented by the Virtual Machines. The heuristic performs simultaneously the dimensioning and routing operations.

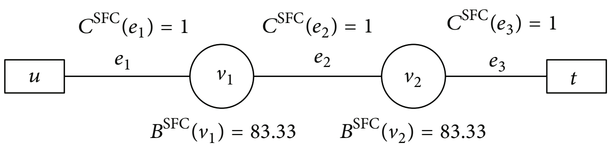
The section is organized as follows. The network and traffic model is introduced in Section 3.1. Sections 3.2 and 3.3 are devoted to illustrating the optimization problem and the proposed heuristic, respectively.

3.1. Network and Traffic Model. Next we introduce the main terminology used to represent the physical network, VNF, and the SFC traffic request [25]. We represent the physical network $\mathcal{PN}$ as a directed graph $\mathcal{G}^{\mathcal{PN}} = (\mathcal{V}^{\mathcal{PN}}, \mathcal{E}^{\mathcal{PN}})$, where $\mathcal{V}^{\mathcal{PN}}$ and $\mathcal{E}^{\mathcal{PN}}$ are the sets of physical nodes and links, respectively. The set $\mathcal{V}^{\mathcal{PN}}$ of nodes is given by the union of the three node sets $\mathcal{V}_{\mathcal{A}}^{\mathcal{PN}}$, $\mathcal{V}_{\mathcal{R}}^{\mathcal{PN}}$, and $\mathcal{V}_{\mathcal{S}}^{\mathcal{PN}}$ that are the sets of access, switching, and server nodes, respectively. The server nodes and links are characterized by the following:

(i) $N_{\text{core}}^{\mathcal{PN}}(w)$: processing capacity of the server node $w \in \mathcal{V}_{\mathcal{S}}^{\mathcal{PN}}$ in terms of the number of cores available;

(ii) $C^{\mathcal{PN}}(d)$: bandwidth of the physical link $d \in \mathcal{E}^{\mathcal{PN}}$.

We assume that F types of VNFs can be provisioned as firewall, IDS, proxy, load balancers, and so on. We denote by $\mathcal{F} = \{f_1, f_2, \dots, f_F\}$ the set of VNFs, with f_i being the



$C^{\mathrm{SFC}}(e_1) = 1$
$C^{\mathrm{SFC}}(e_2) = 1$
$C^{\mathrm{SFC}}(e_3) = 1$
u
e_1
v_1
e_2
v_2
e_3
t
$B^{\mathrm{SFC}}(v_1) = 83.33$
$B^{\mathrm{SFC}}(v_2) = 83.33$

FIGURE 1: An example of graph representing an SFC request for a flow characterized by the bandwidth of 1 Mbps and packet length of 1500 bytes.

ith VNF type. The packet processing time of the VNF f_i is denoted by t_i^{proc} $(i = 1, \dots, F)$.

We also assume that the network operator is receiving T Service Function Chain (SFC) requests known in advance. The ith SFC request is characterized by the graph $\mathscr{G}_i^{\mathscr{SFC}} = (\mathscr{V}_i^{\mathscr{SFC}}, \mathscr{E}_i^{\mathscr{SFC}})$, where $\mathscr{V}_i^{\mathscr{SFC}}$ represents the set of access and VNF nodes and $\mathscr{E}_i^{\mathscr{SFC}}$ $(i = 1, \dots, T)$ denotes the links between them. In particular the set $\mathscr{V}_i^{\mathscr{SFC}}$ is given by the union of $\mathscr{V}_{i,A}^{\mathscr{SFC}}$ and $\mathscr{V}_{i,F}^{\mathscr{SFC}}$ denoting the set of access nodes and VNFs, respectively. The graph is characterized by the following parameters:

(i) α_{vw}: assuming the value 1 if the access node $v \in \bigcup_{i=1}^{T} \mathscr{V}_{i,A}^{\mathscr{SFC}}$ characterizes an SFC request starting/terminating from/to the physical access nodes $w \in \mathscr{V}_{\mathscr{A}}^{\mathscr{PN}}$; otherwise its value is 0;

(ii) β_{vk}: assuming the value 1 if the VNF node $v \in \bigcup_{i=1}^{T} \mathscr{V}_{i,F}^{\mathscr{SFC}}$ needs the application of the VNF f_k $(k \in [1, \dots, F])$; otherwise its value is 0;

(iii) $B^{\mathscr{SFC}}(v)$: the processing capacity requested by the VNF node $v \in \bigcup_{i=1}^{T} \mathscr{V}_{i,F}^{\mathscr{SFC}}$; the parameter value depends on both the packet length and the bandwidth of the packet flow incoming to the VNF node;

(iv) $C^{\mathscr{SFC}}(e)$: bandwidth requested by the link $e \in \bigcup_{i=1}^{T} \mathscr{E}_i^{\mathscr{SFC}}$.

An example of SFC request is represented in Figure 1 where the traffic emitted by the ingress access node u has to be handled by the VNF nodes v_1 and v_2 and it is terminating to the egress access node t. The access and VNF nodes are interconnected by the links e_1, e_2, and e_3. If the flow bandwidth is 1 Mbps and the packet length is 1500 bytes we obtain the processing capacities $B^{\mathscr{SFC}}(v_1) = B^{\mathscr{SFC}}(v_2) = 83.33$ while the bandwidths $C^{\mathscr{SFC}}(e_1)$, $C^{\mathscr{SFC}}(e_2)$, and $C^{\mathscr{SFC}}(e_3)$ of all links equal 1.

3.2. Optimization Problem for the SFC Routing and VNF Instance Dimensioning. The objective of the SFC Routing and VNF Instance Dimensioning (SRVID) problem is to maximize the number of accepted SFC requests. The output of the problem is characterized by the following: (i) the servers in which the VNF nodes are executed and (ii) the network paths in which the virtual links of the SFC are routed. This arrangement has to be accomplished without violating both the server processing and the physical link capacities. We assume that all of the servers create a VNF instance for each type of VNF that will be shared by the SFCs using that server and requesting that type of VNF. For this reason each server will activate F VNF instances, one for each type, and another output of the problem is to determine the number of Vcores to be assigned to each VNF instance.

We assume that one virtual link of the SFC graphs can be routed through single physical network path. We introduce the following notations:

(i) $\mathscr{P}$: set of paths in $\mathscr{G}^{\mathscr{PN}}$,

(ii) δ_{dp}: the binary function assuming value 1 or 0 if the network link d belongs or does not to the path $p \in \mathscr{P}$, respectively,

(iii) $a^{\mathscr{PN}}(p)$ and $b^{\mathscr{PN}}(p)$: origin and destination nodes of the path $p \in \mathscr{P}$,

(iv) $a^{\mathscr{SFC}}(d)$ and $b^{\mathscr{SFC}}(d)$: origin and destination nodes of the virtual link $e \in \bigcup_{i=1}^{T} \mathscr{E}_i^{\mathscr{SFC}}$.

Next we formulate the optimal SRVID problem characterized by the following optimization variables:

(i) x_h: binary variable assuming the value 1 if the hth SFC request is accepted; otherwise its value is zero;

(ii) y_{wk}: integer variable characterizing the number of Vcores allocated to the VNF instance of type k in the server $w \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}$;

(iii) z_{vw}^{k}: binary variable assuming the value 1 if the VNF node $v \in \bigcup_{i=1}^{T} \mathscr{V}_{i,F}^{\mathscr{SFC}}$ is served by the VNF instance of type k in the server $w \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}$;

(iv) u_{dp}: binary variable assuming the value 1 if the virtual link $e \in \bigcup_{i=1}^{T} \mathscr{E}_i^{\mathscr{SFC}}$ is embedded in the physical network path $p \in \mathscr{P}$; otherwise its value is zero.

Next we report the constraints for the optimization variables:

$$\sum_{k=1}^{F} y_{wk} \le N_{\mathrm{core}}^{\mathscr{PN}}(w), \qquad w \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}, \tag{1}$$

$$\sum_{k=1}^{F} \sum_{w \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}} z_{vw}^{k} \le 1, \quad v \in \bigcup_{i=1}^{T} \mathscr{V}_{i,F}^{\mathscr{SFC}}, \tag{2}$$

$$z_{vw}^{k} \le \beta_{vk}, \qquad k \in [1, \dots, F], \; v \in \bigcup_{i=1}^{T} \mathscr{V}_{i,F}^{\mathscr{SFC}}, \; w \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}, \tag{3}$$

$$\sum_{v \in \bigcup_{i=1}^{T} \mathscr{V}_{i,F}^{\mathscr{SFC}}} z_{vw}^{k} B^{\mathscr{SFC}}(v)\, t_k^{\mathrm{proc}} \le y_{wk}, \qquad k \in [1, \dots, F], \; w \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}, \tag{4}$$

$$u_{dp} \leq z^{k}_{a^{\mathcal{SFC}}(d)a^{\mathcal{PN}}(p)},$$
$$a^{\mathcal{SFC}}(d) \in \bigcup_{i=1}^{T} \mathcal{V}^{\mathcal{SFC}}_{i,F}, \ k \in [1,\dots,F], \ p \in \mathcal{P}, \tag{5}$$

$$u_{dp} \leq z^{k}_{b^{\mathcal{SFC}}(d)b^{\mathcal{PN}}(p)},$$
$$b^{\mathcal{SFC}}(d) \in \bigcup_{i=1}^{T} \mathcal{V}^{\mathcal{SFC}}_{i,F}, \ k \in [1,\dots,F], \ p \in \mathcal{P}, \tag{6}$$

$$u_{dp} \leq \alpha^{k}_{a^{\mathcal{SFC}}(d)a^{\mathcal{PN}}(p)},$$
$$a^{\mathcal{SFC}}(d) \in \bigcup_{i=1}^{T} \mathcal{V}^{\mathcal{SFC}}_{i,A}, \ k \in [1,\dots,F], \ p \in \mathcal{P}, \tag{7}$$

$$u_{dp} \leq \alpha^{k}_{b^{\mathcal{SFC}}(d)b^{\mathcal{PN}}(p)},$$
$$b^{\mathcal{SFC}}(d) \in \bigcup_{i=1}^{T} \mathcal{V}^{\mathcal{SFC}}_{i,A}, \ k \in [1,\dots,F], \ p \in \mathcal{P}, \tag{8}$$

$$\sum_{p\in\mathcal{P}} u_{dp} \leq 1, \quad d \in \bigcup_{i=1}^{T} \mathcal{E}^{\mathcal{SFC}}_{i}, \tag{9}$$

$$\sum_{d\in\bigcup_{i=1}^{T}\mathcal{E}^{\mathcal{SFC}}_{i}} C^{\mathcal{SFC}}(d) \sum_{p\in\mathcal{P}} \delta_{ep} u_{dp} \leq C^{\mathcal{PN}}(e), \tag{10}$$

$$x_h \leq \sum_{k=1}^{F} \sum_{w\in\mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}} z^{k}_{vw},$$
$$h \in [1,\dots,T], \ v \in \mathcal{V}^{\mathcal{SFC}}_{h,F}, \tag{11}$$

$$x_h \leq \sum_{p\in\mathcal{P}} u_{dp},$$
$$h \in [1,\dots,T], \ d \in \mathcal{E}^{\mathcal{SFC}}_{h}. \tag{12}$$

Constraint (1) establishes the fact that at most a number of Vcores equal to the available ones are used for each server node $w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}$. Constraint (2) expresses the condition that a VNF can be served by one only VNF instance. To guarantee that a node $v \in \mathcal{V}^{\mathcal{SFC}}_{h,F}$ needing the application of a VNF type is mapped to a correct VNF instance we introduce constraint (3). Constraint (4) limits the number of VNF nodes assigned to one VNF instance by taking into account both the number of Vcores assigned to the VNF instance and the required VNF node processing capacities. Constraints (5)–(8) establish the fact that when the virtual link d is supported by the physical network path p then $a^{\mathcal{PN}}(p)$ and $b^{\mathcal{PN}}(p)$ must be the physical network nodes that the nodes $a^{\mathcal{SFC}}(d)$ and $b^{\mathcal{SFC}}(d)$ of the virtual graph are assigned to. The choice of mapping of any virtual link on a single physical network path is represented by constraint (9). Constraint (10) avoids any overloading on any physical network link. Finally constraints (11) and (12) establish the fact that an SFC request can be accepted when the nodes and the links of the SFC graph are assigned to VNF instances and physical network paths.

The problem objective is to maximize the number of SFCs accepted given by

$$\max \sum_{h=1}^{T} x_h. \tag{13}$$

The introduced optimization problem is intractable because it requires solving a NP-hard bin packing problem [26]. Due to its high complexity, it is not possible to solve the problem directly in a timely manner given the large number of servers and network nodes. For this reason we propose an efficient heuristic in Section 3.3.

3.3. Maximizing the Accepted SFC Requests Number (MASRN) Heuristic. The Maximizing the Accepted SFC Requests Number (MASRN) heuristic is based on the embedding algorithm proposed in [14] and has the objective of maximizing the number of accepted SFC requests. The MASRN algorithm tries routing the SFCs one by one by using the least loaded link and server resources so as to balance their use. Algorithm 1 illustrates the operation mode of the MASRN algorithm. The routing of a target SFC, the k_{tar}th, is illustrated. The main inputs of the algorithm are (i) the set T_{pre} containing the indexes of the SFC requests routed successfully before the k_{tar}th SFC request; (ii) the k_{tar}th SFC request graph $\mathcal{G}^{\mathcal{SFC}}_{k_{\text{tar}}} = (\mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}}}, \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}})$; (iii) the values of the parameters α_{vw} for the k_{tar}th SFC request; (iv) the values of the variables z^{k}_{vw} and u_{dp} for the SFC requests successfully routed before the k_{tar}th SFC request and defined in Section 3.2.

The operation mode of the heuristic is based on the following variables:

(i) $A^{k}(w)$: the load of the cores allocated for the VNF instance of type $k \in [1,\dots,F]$ in the server $w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}$; the value of $A^{k}(w)$ is initialized by taking into account the core resource amount used by the SFCs successfully routed before the k_{tar}th SFC request; hence its initial value is

$$A^{k}(w) = \sum_{v\in\bigcup_{i\in T_{k_{\text{tar}}}}\mathcal{V}^{\mathcal{SFC}}_{i,F}} z^{k}_{vw} B^{\mathcal{SFC}}(v)\, t^{\text{proc}}_{k}. \tag{14}$$

(ii) $S_N(w)$: defined as the stress of the node $w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}$ [14] and characterizing the server load; its value is initialized to

$$S_N(w) = \sum_{k=1}^{F} A^{k}(w). \tag{15}$$

(iii) $S_L(e)$: defined as the stress of the physical network link $e \in \mathcal{E}^{\mathcal{PN}}$ [14] and characterizing the link load; its value is initialized to

$$S_L(e) = \sum_{d\in\bigcup_{i\in T_{k_{\text{tar}}}}\mathcal{E}^{\mathcal{SFC}}_{i}} C^{\mathcal{SFC}}(d) \sum_{p\in\mathcal{P}} \delta_{ep} u_{dp}. \tag{16}$$

```
(1) Input: Physical Network Graph $\mathcal{G}^{\mathcal{PN}} = (\mathcal{V}^{\mathcal{PN}}, \mathcal{E}^{\mathcal{PN}})$; $k_{\text{tar}}$th SFC request; $k_{\text{tar}}$th SFC request Graph
    $\mathcal{G}^{\mathcal{SFC}}_{k_{\text{tar}}} = (\mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}}}, \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}})$; $T_{\text{pre}}$;
    $\{\alpha_{vw},\ v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},A}\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{A}}\}$;
    $\{z^k_{vw},\ k \in [1,\ldots,F]\ v \in \bigcup_{i \in T_{\text{pre}}} \mathcal{V}^{\mathcal{SFC}}_{i,F}\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}\}$;
    $\{u_{dp},\ d \in \bigcup_{i \in T_{\text{pre}}} \mathcal{E}^{\mathcal{SFC}}_{i}\ p \in \mathcal{P}\}$;
(2) Variables: $\{A^k(w),\ k \in [1,\ldots,F]\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}\}$;
    $\{S_N(w),\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}\}$;
    $\{S_L(e),\ e \in \mathcal{E}^{\mathcal{PN}}\}$;
    $\{y_{wk},\ k \in [1,\ldots,F]\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}\}$;
    /* Evaluation of the candidate mapping of $\mathcal{G}^{\mathcal{SFC}}_{k_{\text{tar}}} = (\mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}}}, \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}})$ in $\mathcal{G}^{\mathcal{PN}} = (\mathcal{V}^{\mathcal{PN}}, \mathcal{E}^{\mathcal{PN}})$ */
(3) assign the node $v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},A}$ to the physical network node $w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}$ according to the value of the parameter $\alpha_{vw}$;
(4) select the nodes $w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}$ and the physical network paths $p \in \mathcal{P}$ to be assigned to the server nodes $v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},F}$ and virtual links
    $d \in \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}},S}$ by applying the algorithm proposed in [14] and based on the values of the node and link stresses $S_N(w)$ and $S_L(e)$;
    determine the variable values:
    $\{z^k_{vw},\ k \in [1,\ldots,F]\ v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},F}\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}\}$;
    $\{u_{dp},\ d \in \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}}\ p \in \mathcal{P}\}$;
    /* Server Resource Availability Check Phase */
(5) for $v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},F}, w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}, k \in [1,\ldots,F]$ do
(6)     if $\sum_{s=1 (s \neq k)}^{F} y_{ws} + \lceil A^k(w) + z^k_{vw} B^{\mathcal{SFC}}(v) t^{\text{proc}}_k \rceil \leq N^{\mathcal{PN}}_{\text{core}}(w)$ then
(7)         $A^k(w) = A^k(w) + z^k_{vw} B^{\mathcal{SFC}}(v) t^{\text{proc}}_k$;
(8)         $S_N(w) = S_N(w) + z^k_{vw} B^{\mathcal{SFC}}(v) t^{\text{proc}}_k$;
(9)         $y_{wk} = \lceil A^k(w) + z^k_{vw} B^{\mathcal{SFC}}(v) t^{\text{proc}}_k \rceil$;
(10)    else
(11)        REJECT the $k_{\text{tar}}$th SFC request
(12)    end if
(13) end for
    /* Link Resource Availability Check Phase */
(14) for $e \in \mathcal{E}^{\mathcal{PN}}$ do
(15)    if $S_L(e) + \sum_{d \in \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}}} C^{\mathcal{SFC}}(d) \sum_{p \in \mathcal{P}} \delta_{ep} u_{dp} \leq C^{\mathcal{PN}}(e)$ then
(16)        $S_L(e) = S_L(e) + \sum_{d \in \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}}} C^{\mathcal{SFC}}(d) \sum_{p \in \mathcal{P}} \delta_{ep} u_{dp}$;
(17)    else
(18)        REJECT the $k_{\text{tar}}$th SFC request
(19)    end if
(20) end for
(21) Output: $\{z^k_{vw},\ k \in [1,\ldots,F]\ v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},F}\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}\}$;
    $\{u_{dp},\ d \in \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}}\ p \in \mathcal{P}\}$
```

Algorithm 1: MASRN algorithm.

The candidate physical network links and nodes in which to embed the target SFC are evaluated. First of all the access nodes are evaluated (line 3) according to the values of the parameters α_{vw}, $v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},A}$, $w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{A}}$. Next the MASRN algorithm selects a cluster of server nodes (line 4) that are not lightly loaded but also likely to result in low substrate link stresses when they are connected. Details on the procedure are reported in [14].

In the next phases the resource availability in the selected cluster is checked. The resource availability in the server nodes and physical network links is verified in the Server (lines 5–13) and Link (lines 14–20) Resource Availability Check Phases, respectively. If resources are not available in either links or nodes, the target SFC request is rejected. In particular, in the Server Resource Availability Check Phase, the algorithm verifies whether the allocation of new Vcores is needed and in this case their availability is checked (line 6). The variable y_{wk} is also updated in this phase (line 9).

Finally the outputs (line 21) of the MASRN algorithm are the selected values for the variables $\{z^k_{vw}, k \in [1,\ldots,F]\ v \in \mathcal{V}^{\mathcal{SFC}}_{k_{\text{tar}},F}\ w \in \mathcal{V}^{\mathcal{PN}}_{\mathcal{S}}\}$ and $\{u_{dp}, d \in \mathcal{E}^{\mathcal{SFC}}_{k_{\text{tar}}}\ p \in \mathcal{P}\}$.

The computational complexity of the MASRN algorithm depends on the procedure for the evaluation of the potential nodes [14]. Let N_s be the total number of servers. The complexity of the phase in which the potential nodes are evaluated can be carried out according to the following remarks: (i) as many servers as the number of the VNFs of the SFC are determined and (ii) the hth server is selected by evaluating the $(N_s - h)$ shortest paths and by performing a minimum operation of a list with $(N_s - h)$ elements. According to these remarks the computational complexity is given by $\mathrm{O}(V(V + N_s)N_l \log(N_s + N_n))$, where V is the

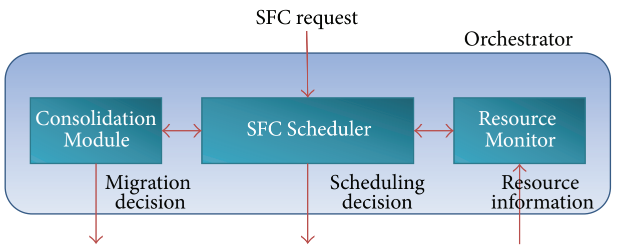


Figure 2: SFC planner architecture.

maximum number of VNFs in an SFC and N_l and N_n are the numbers of nodes and links of the network.

4. Online Algorithms for SFC Routing in NFV Network Architectures

In the online approach the SFC requests arrive at the system over the time and each of them is characterized by a certain duration. In order to reduce the complexity of the online SFC resource allocation algorithm we designed an SFC planner whose general architecture is described in Section 4.1. The operation mode of the SFC planner is based on some algorithms that we describe in Section 4.2.

4.1. SFC Planner Architecture. The architecture of the SFC planner is shown in Figure 2. It could make up the main component of an orchestrator in the NFV architecture [6]. It is composed of three components: the *SFC Scheduler*, the *Resource Monitor*, and the *Consolidation Module*.

Once the SFC Scheduler receives an SFC request, it executes an algorithm to verify if resources are available and to decide which resources to use.

The physical resource, as well as the Virtual Machines running on the servers, is monitored by the *Resource Monitor*. If a failure of any physical/virtual node or link occurs it notifies the event to the SFC Scheduler.

The Consolidation Module allows for the server resource consolidation in low traffic periods. When the consolidation technique is applied, VMs are migrated to as fewer servers as possible; the remaining ones are turned off and power consumption saving can be achieved.

4.2. SFC Scheduling and Consolidation Algorithms. In this section we describe the algorithms executed by the SFC Scheduler and the Consolidation Module.

Whenever a new SFC request arrives at the SFC planner, the SFC scheduling algorithm is executed in order to find the best embedding in the system, maintaining a uniform usage of the network resource. The main steps of this procedure are reported in the flow chart in Figure 3 and are based on MASRN heuristic described in Section 3.3. The algorithm starts selecting the set of servers on which the VNFs requested by the SFC will be executed. In the second step the physical paths on which the virtual links will be mapped are selected. If there are not enough available

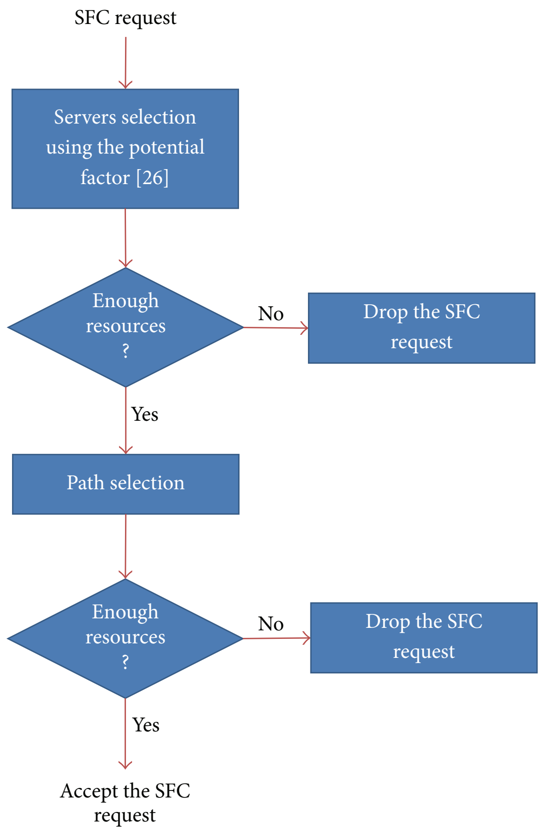


Figure 3: Flow chart of the SFC scheduling algorithm.

resources in the first or in the second step the request is dropped.

The flow chart of the consolidation algorithm is reported in Figure 4. Each server $s \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}$ is characterized by a parameter η_s defined as the ratio of the total amount of incoming/outgoing traffic Λ_s that it is handling to the power P_s it is consuming; that is, $\eta_s = \Lambda_s/P_s$. The power consumption model assumes a constant power contribution and a variable one that linearly increases versus the handled traffic. The server power consumption is expressed by

$$P_s = P_{\text{idle}} + (P_{\max} - P_{\text{idle}})\frac{L_s}{C_s}, \quad \forall s \in \mathscr{V}_{\mathscr{S}}^{\mathscr{PN}}, \tag{17}$$

where P_{idle} is the power consumed by the server when no traffic is handled, $P_{\max}$ is the maximum server power consumption, L_s is the total load handled by the server s, and C_s is its maximum capacity. The maximum power that the server can consume is expressed as $P_{\max} = P_{\text{idle}}/a$, where a is a parameter characterizing how much the power consumption is depending on the handled traffic. The power consumption is as much more rate adaptive as a is lower. For $a = 1$ the rate adaptive power consumption is absent and the consumed power is constant.

The proposed algorithm tries switching off the server with minimum power consumption per bit carried. For this reason when the consolidation algorithm is executed it starts selecting the server $s_{\min}$ with the minimum value of η_s as

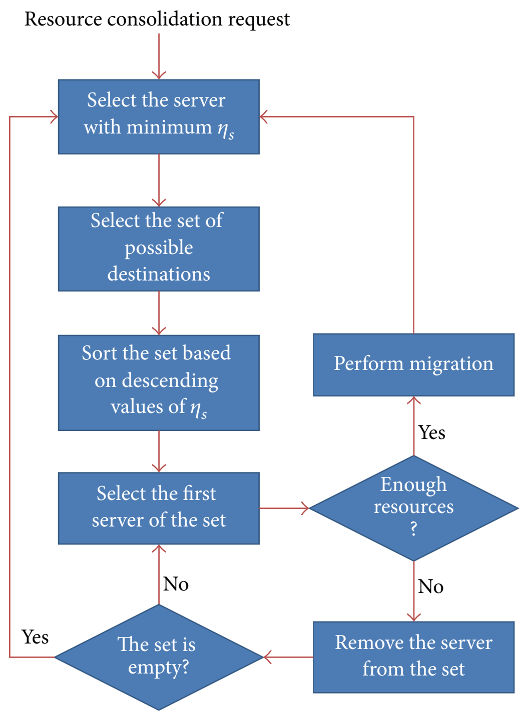

Resource consolidation request
Select the server with minimum η_s
Select the set of possible destinations
Sort the set based on descending values of η_s
Select the first server of the set
Enough resources ?
Yes
Perform migration
No
Remove the server from the set
The set is empty?
No
Yes

FIGURE 4: Flow chart of the SFC consolidation algorithm.

the one to turn off. A set of possible destination servers able to host the VMs executed on $s_{\min}$ is then evaluated and sorted in descending order based again on the value of η_s. The algorithm proceeds selecting the first element of the ordered set and evaluating if there are enough resources in the site to reroute all the flows generated from or terminated to $s_{\min}$. If it succeeds the migration is performed and the server $s_{\min}$ is turned off. If it fails the destination server is removed and the next server of the list is considered. When the ordered set of possible destinations becomes empty another server to turn off is selected.

The SFC Consolidation Module also manages the resource deconsolidation procedure when the reactivation of physical servers/links is needed. Basically the deconsolidation algorithm selects the most loaded VMs already migrated to a new server and moves them to their original servers. The flows handled by these VMs are accordingly rerouted.

5. Numerical Results

We evaluate the effectiveness of the proposed algorithms in the case of the network scenario reported in Figure 5. The network is composed of five core nodes, five edge nodes, and six access nodes in which the SFC requests are randomly generated and terminated. A Network Function Virtualization (NFV) site is connected to edge nodes 3 and 4 through two access routers 1 and 2. The NFV site is also composed of two switches and eight servers. In the basic configuration, 40 Gbps links are considered except the links connecting the server to the switches in the NFV sites whose rate is equal to 10 Gbps. The sixteen servers are equipped with 48 Vcores each.

Each SFC request is composed of three Virtual Network Functions (VNFs), that is, a firewall, a load balancer, and VPN encryption whose packet processing times are assumed to be equal to 7,08 μs, 0,65 μs, and 1,64 μs [27], respectively. We also assume that the load balancer splits the input traffic towards a number of output links chosen uniformly from 1 to 3.

An example of graph for a generated SFC is reported in Figure 6 in which u_t is an ingress access node and v_t^1, v_t^2, and v_t^3 are egress access nodes. The first and second VNFs are a firewall and VPN encryption; the third VNF is a load balancer splitting the traffic towards three output links. In the considered case study we also assume that the SFC handles traffic flows of packet length equal to 1500 bytes and required bandwidth uniformly distributed from 500 Mbps to 1 Gbps.

5.1. Offline SFC Scheduling. In the offline case we assume that a given number T of SFC requests are a priori known. For the case study previously described we apply the MASRN heuristic proposed in Section 3.3. We report the percentage of the dropped SFCs in Figure 7 as a function of the offered number of SFCs. We report the curve for the basic scenario and the ones in which the link capacities are doubled, quadrupled, and increased per ten times, respectively. From Figure 7 we can see how, in order to have a dropped SFC percentage lower than 10%, the number of SFCs requests has to be smaller than 60 in the case of basic scenario. This remarkable blocking is due to the shortage of network capacity. In fact we can notice from Figure 7 how the dropped SFC percentage significantly decreases when the link bandwidth increases. For instance, when the capacity is quadrupled, the network infrastructure is able to accommodate up to 180 SFCs without any loss.

This is confirmed in Figure 8 where we report the number of Vcores occupied in the servers in the case of $T = 360$. Each of the servers is represented on the x-axis with the identification (ID) of Figure 5. We also show in Figure 8 the number of Vcores allocated to each firewall, load balancer, and VPN encryption VNF with the blue, green, and red colors, respectively. The basic scenario case and the ones in which the link capacity is doubled, quadrupled, and increased by 10 times are reported in Figures 8(a), 8(b), 8(c), and 8(d), respectively. From Figure 8 we can notice how the MASRN heuristic works correctly by guaranteeing a uniform occupancy of the servers in terms of total number of Vcores used. We can also notice how the firewall VNF is the one needing the higher number of Vcores allocated. That is a consequence of the higher processing time that the firewall VNF requires with respect to the load balancer and VPN encryption VNFs.

5.2. Online SFC Scheduling. The numerical results are achieved in the following traffic scenario. The traffic pattern we consider is supposed to be sinusoidal with a peak value during daytime and minimum value during nighttime. We also assume that SFC requests follow a Poisson process and the SFC duration time is distributed according to an

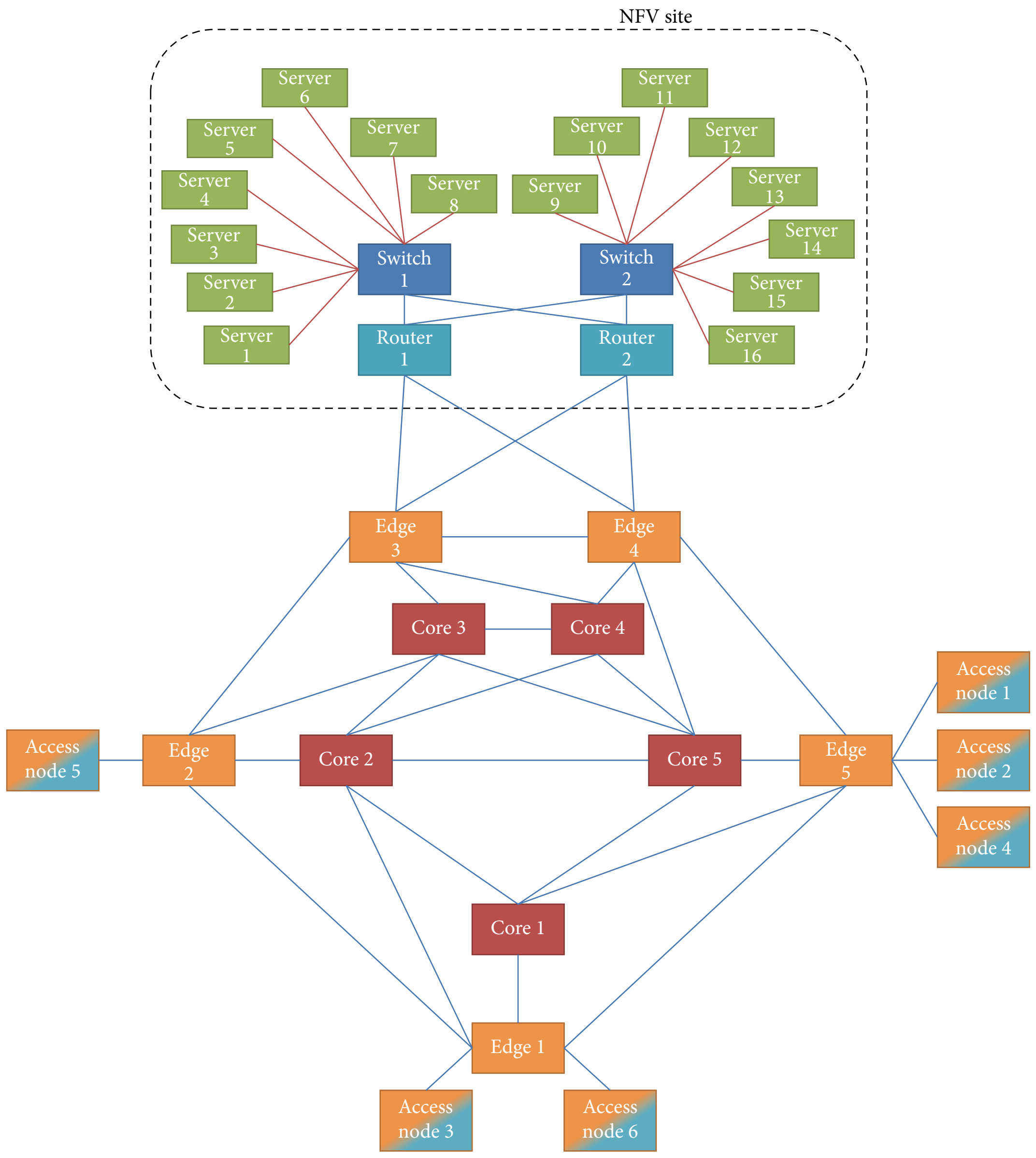


Figure 5: The network topology is composed of six access nodes, five edge nodes, and five core nodes. The NFV site is composed of two routers, two switches, and sixteen servers.

exponential distribution. The following parameters define the SFC requests in the Peak Hour Interval (PHI):

(i) λ is the arrival rate of SFC requests;

(ii) μ is the termination rate of an embedded SFC;

(iii) $[\beta^{\min}, \beta^{\max}]$ is the range in which the bandwidth request for an SFC is randomly selected.

We consider K time intervals in a day and each of them is characterized by a scaling factor α_h with $h = 0, \ldots, K-1$. In order to capture the sinusoidal shape of the traffic in a day, α_h is evaluated with the following expression:

$$\alpha_h = \begin{cases} 1, & \text{if } h = 0, \\ 1 - 2\dfrac{h}{K}\left(1 - \alpha_{\min}\right), & h = 1, \ldots, \dfrac{K}{2}, \\ 1 - 2\dfrac{K-h}{K}\left(1 - \alpha_{\min}\right), & h = \dfrac{K}{2} + 1, \ldots, K-1, \end{cases} \tag{18}$$

where α_0 and $\alpha_{\min}$ refer to the peak traffic and the minimum traffic scenario, respectively.

The parameter α_h affects the SFC arrival rate and the bandwidth requested. In particular we have the following

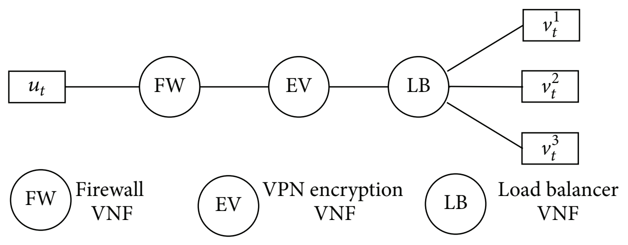


Figure 6: An example of SFC composed of one firewall VNF, one VPN encryption VNF, and one load balancer VNF that splits the input traffic towards three output logical links. u_t denotes the ingress access node while v_t^1, v_t^2, and v_t^3 denote the egress access nodes.

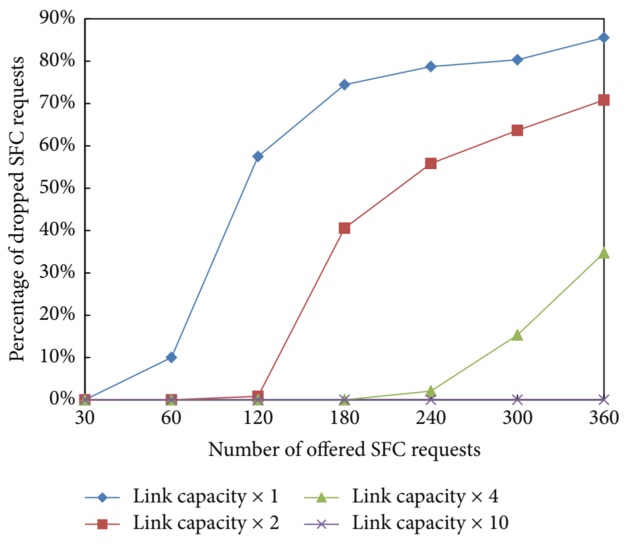


Figure 7: Dropped SFC percentage as a function of the number of SFCs offered. The four curves are reported for the basic scenario case and the ones in which the link capacities are doubled, quadrupled, and increased by 10 times.

expression for the SFC request rate λ_h and the minimum and the maximum requested bandwidths $\beta_h^{\min}$ and $\beta_h^{\max}$, respectively, in the interval h $(h = 0, \ldots, K-1)$:

$$\lambda_h = \alpha_h \lambda,$$
$$\beta_h^{\min} = \alpha_h \beta^{\min}, \tag{19}$$
$$\beta_h^{\max} = \alpha_h \beta^{\max}.$$

Next we evaluate the performance of SFC planner proposed in Section 4 in dynamic traffic scenario and for parameter values $\beta^{\min} = 500$ Mbps, $\beta^{\max} = 1$ Gbps, $\lambda = 1$ rich/min, and $\mu = 1/15\ \mathrm{min}^{-1}$. We also assume $K = 24$ with traffic change and consequently application of the consolidation algorithm every hour.

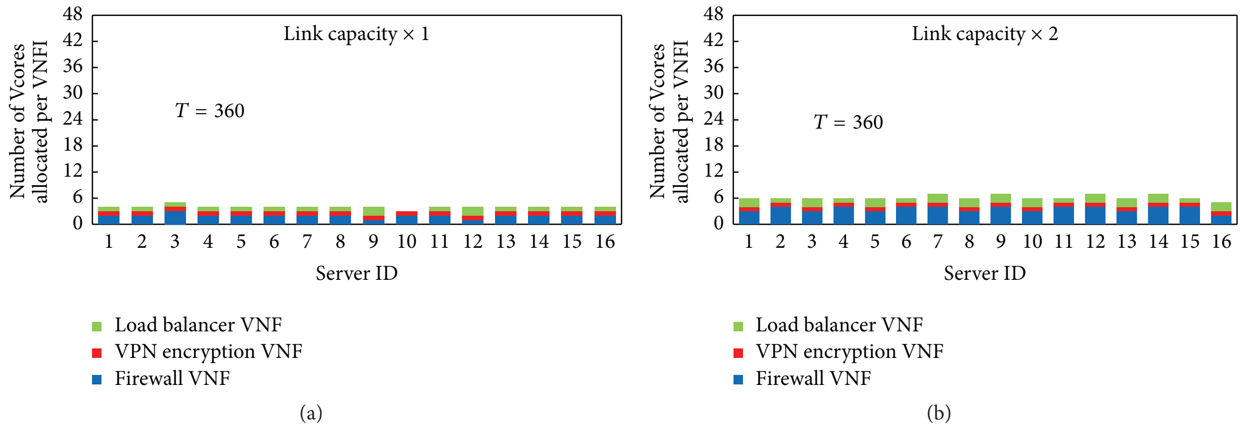
Finally we consider servers whose power consumption is characterized by the expression (17) with parameters $P_{\max} = 1000$ W and $C_s = 10$ Gbps.

We report the SFC blocking probability as a function of the average number of offered SFC requests during the PHI (λ/μ) in Figure 9. We show the results in the case of basic link capacity scenario and in the ones obtained considering a capacity of the links that is twice and four times higher than the basic one. We consider the case of server with no rate adaptive power consumption $(a = 1)$. As we can observe, when the offered traffic is low, the degradation in terms of SFC blocking probability introduced by the consolidation technique increases. In fact in this low traffic scenario, more resource consolidation is possible with the consequence of higher SFC blocking probabilities. When the offered traffic increases the blocking probability with or without applying the consolidation technique does not change much because the network is heavily loaded and only few servers can be turned off. Furthermore, as we can expect, the blocking probability decreases when the links capacity increases. The performance loss due to the application of the consolidation technique is what we have to pay in order to obtain benefits in terms of power saved by turning off servers. The curves in Figure 10 compare the power consumption percentage savings that we can achieve in the cases $a = 0.1$ and $a = 1$. The results show that when the offered traffic decreases and the link capacity increases, higher power consumption saving can be achieved. The reason is that we have more resources on the links and less loaded VMs that allow for a higher number of VM migrations and resource consolidation. The other result to notice is that the use of rate adaptive servers reduces the power consumption saving when we perform resource consolidation. This is due to the lower value P_{idle} of the constant power consumption of the server that leads to lower power consumption saving when a server is switched off.

6. Conclusions

The aim of this paper has been to propose heuristics for the resource dimensioning and the routing of Service Function Chain in network architectures employing the Network Function Virtualization technology. We have introduced the optimization problem that has the objective of minimizing the number of SFCs offered and the compliance of server processing and link bandwidth capacity. With the problem being NP-hard, we have proposed the Maximizing the Accepted SFC Requests Number (MASRN) heuristic that is based on the uniform occupancy of the server and link resources. The heuristic is also able to dimension the Vcores of the servers by evaluating the number of Vcores to be allocated to each VNF instance.

An SFC planner has been already proposed for the dynamic traffic scenario. The planner is based on a consolidation algorithm that allows for the switching off of servers during the low traffic periods. A case study has been analyzed in which we have shown that the heuristics works correctly by uniformly allocating the server resources and reserving a higher number of Vcores for the VNFs characterized by higher packet processing time. Furthermore the proposed SFC planner allows for a power consumption saving dependent on the offered traffic and varying from 10% to 70%. Such a saving is to be paid with an increase in SFC blocking probability. As future research we will propose and investigate Virtual Machine migration policies



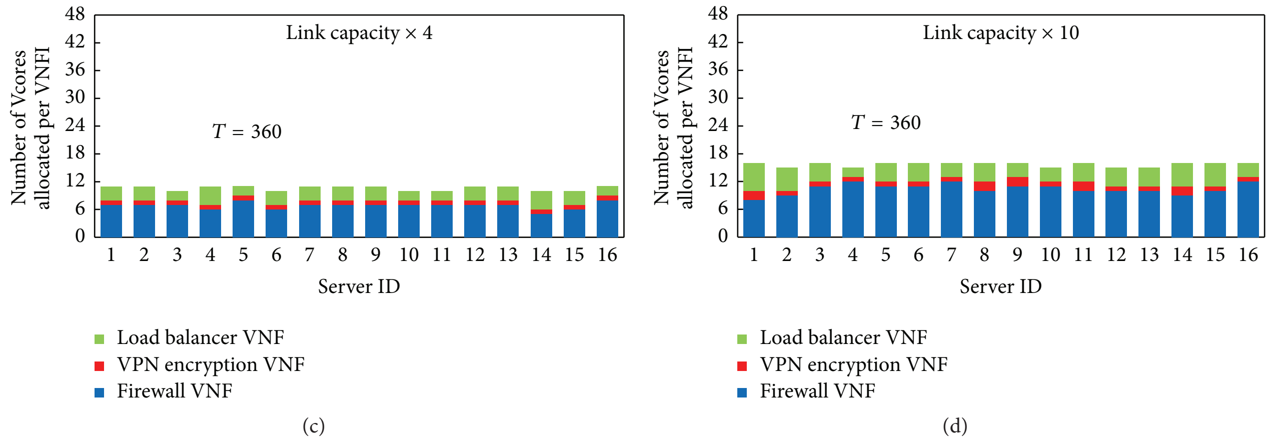


FIGURE 8: Number of allocated Vcores in each server. The number of Vcores for firewall, load balancer, and encryption VPN VNFs are represented with blue, green, and red colors.

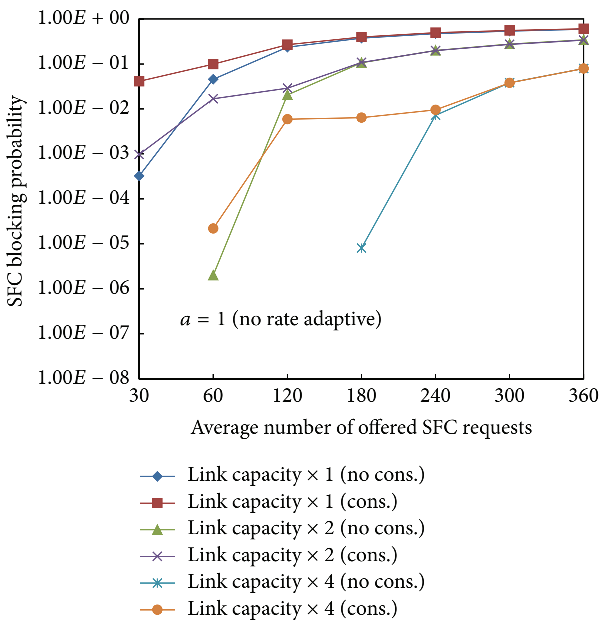


FIGURE 9: SFC blocking probability as a function of the average number of SFCs offered in the PHI for the cases in which the consolidation technique is applied and it is not. The server power consumption is constant ($a = 1$).

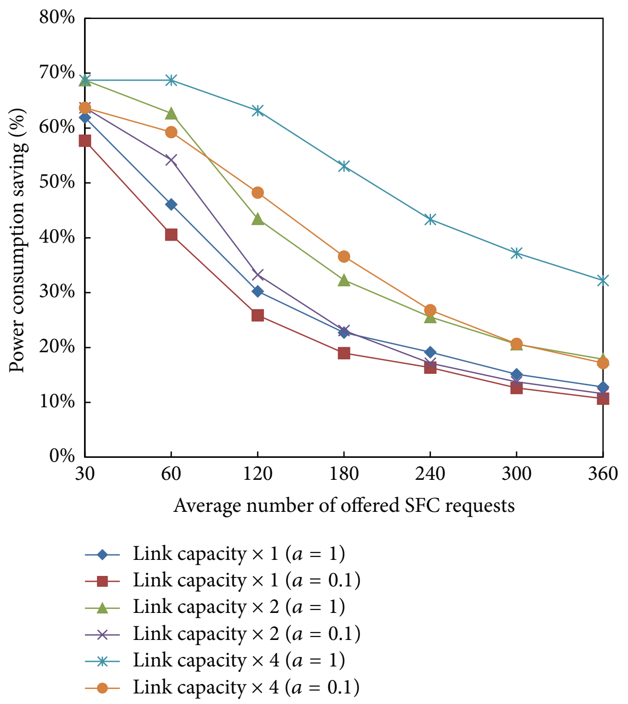


FIGURE 10: The power consumption percentage saving achieved with the application of the consolidation technique versus the average number of SFCs offered in the PHI. The cases $a = 1$ and $a = 0.1$ are considered.

allowing for the achievement of a right trade-off between power consumption saving and SFC blocking probability degradation.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The work described in this paper has been funded by Telecom Italia within the research agreement titled "Definition and Evaluation of Protocols and Architectures of a Network Automation Platform for TI-NVF Infrastructure."

References

[1] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[2] P. Veitch, M. J. McGrath, and V. Bayon, "An instrumentation and analytics framework for optimal and robust NFV deployment," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 126–133, 2015.

[3] R. Yu, G. Xue, V. T. Kilari, and X. Zhang, "Network function virtualization in the multi-tenant cloud," *IEEE Network*, vol. 29, no. 3, pp. 42–47, 2015.

[4] Z. Bronstein, E. Roch, J. Xia, and A. Molkho, "Uniform handling and abstraction of NFV hardware accelerators," *IEEE Network*, vol. 29, no. 3, pp. 22–29, 2015.

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[6] ETSI Industry Specification Group (ISG) NFV, "ETSI Group Specifications on Network Function Virtualization. 1st Phase Documents," January 2015, http://docbox.etsi.org/ISG/NFV/Open/.

[7] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vepc)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, 2014.

[8] The Internet Engineering Task Force (IETF), *Service Function Chaining (SFC) Working Group (WG)*, 2015, https://datatracker.ietf.org/wg/sfc/charter/.

[9] The Internet Engineering Task Force (IETF), Service Function Chaining (SFC) Working Group (WG), Documents, 2015, https://datatracker.ietf.org/wg/sfc/documents/.

[10] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku, "MORSA: a multi-objective resource scheduling algorithm for NFV infrastructure," in *Proceedings of the 16th Asia-Pacific Network Operations and Management Symposium (APNOMS '14)*, pp. 1–6, Hsinchum, Taiwan, September 2014.

[11] H. Moens and F. D. Turck, "Vnf-p: a model for efficient placement of virtualized network functions," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM '14)*, pp. 418–423, November 2014.

[12] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft '15)*, pp. 1–9, University College London, London, UK, April 2015.

[13] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS '14)*, pp. 1–9, Krakow, Poland, May 2014.

[14] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, IEEE, Barcelona, Spain, April 2006.

[15] G. Lee, M. Kim, S. Choo, S. Pack, and Y. Kim, "Optimal flow distribution in service function chaining," in *Proceedings of the 10th International Conference on Future Internet (CFI '15)*, pp. 17–20, ACM, Seoul, Republic of Korea, June 2015.

[16] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: a survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[17] M. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM '13)*, pp. 177–184, Ghent, Belgium, May 2013.

[18] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications and Challenges (AllThingsCellular '14)*, pp. 33–38, ACM, Chicago, Ill, USA, August 2014.

[19] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '14)*, pp. 2402–2407, IEEE, Istanbul, Turkey, April 2014.

[20] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proceedings of the IEEE 3rd International Conference on Cloud Networking (CloudNet '14)*, pp. 7–13, October 2014.

[21] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," in *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft '15)*, pp. 1–9, London, UK, April 2015.

[22] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for NFV chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, Article ID 7005460, pp. 1565–1570, 2015.

[23] O. Hyeonseok, Y. Daeun, C. Yoon-Ho, and K. Namgi, "Design of an efficient method for identifying virtual machines compatible with service chain in a virtual network environment," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 11, no. 9, Article ID 197208, 2014.

[24] W. Cerroni and F. Callegati, "Live migration of virtual network functions in cloud-based edge networks," in *Proceedings of the IEEE International Conference on Communications (ICC '14)*, pp. 2963–2968, IEEE, Sydney, Australia, June 2014.

[25] P. Quinn and J. Guichard, "Service function chaining: creating a service plane via network service headers," *Computer*, vol. 47, no. 11, pp. 38–44, 2014.

[26] M. F. Zhani, Q. Zhang, G. Simona, and R. Boutaba, "VDC Planner: dynamic migration-aware Virtual Data Center embedding for clouds," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM '13)*, pp. 18–25, May 2013.

[27] M. Dobrescu, K. Argyraki, and S. Ratnasamy, "Toward predictable performance in software packet-processing platforms," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation*, pp. 1–14, April 2012.