QoS-driven Function Placement Reducing Expenditures in NFV Deployments

Petra Vizarreta*, Massimo Condoluci†, Carmen Mas Machuca*, Toktam Mahmoodi†, and Wolfgang Kellerer*
* Chair of Communication Networks, Technical University of Munich, Germany
petra.vizarreta@lkn.ei.tum.de, {cmas, wolfgang.kellerer }@tum.de
† Department of Informatics, King's College London, UK
{massimo.condoluci, toktam.mahmoodi }@kcl.ac.uk

***Abstract*—With Network Function Virtualization (NFV), network functions are deployed as modular software components on the commodity hardware, and can be further chained to provide services. Network operators offer different classes of services to their users and their requirements are specified in Service Level Agreements (SLA) which include several QoS performance parameters such as the maximum tolerated delay or the minimum availability. So far, state of the art solutions for NFV deployment focus on delay related requirements. However, service availability, which is an important requirement for any SLA is mostly neglected. This paper focuses on the placement of virtualized network functions, with the target to support service differentiation in terms of delay and availability while minimizing the associated costs. We present two solutions: an ILP formulation and an efficient heuristic to obtain near optimal solution. Considering a national core network case study, we show that the proposed function placement solutions are able to guarantee both delay and availability requirements, and imply only a limited increase of used network resources, compared to solutions that only address a single requirement. Finally, we show that the execution time of the proposed heuristic scales well with the size of the problem.**

I. Introduction

Network Function Virtualization (NFV) is a novel network architecture concept where network functions, such as firewalls, Network Address Translation (NAT) or Intrusion Detection and Prevention Systems (IDPS), which are traditionally implemented as specialized hardware, are replaced with software components deployed on commodity hardware [1]. Service Function Chaining, sometimes referred to simply as "service chaining", describes how the network functions can be stitched together to compose a service [2], e.g. voice or video conferencing. Service chaining with virtualized network functions offer to network operators greater flexibility in the service provisioning and lower required resources. First studies of such benefits have been explored in various use cases ranging from mobile and fixed access to Content Delivery Networks (CDN) [1].

Network operators offer a wide service portfolio. The details of each service to a particular customer are formalized in the Service Level Agreements (SLAs). Although the SLAs may vary among operators, they typically contain QoS parameters such as minimum guaranteed bit rate, maximum delay, port availability and packet loss [3] [4]. In order to increase the operator's ability to fulfil the SLA requirements, it is required to define comprehensive models of end-to-end QoS of the service chains in NFV based networks.

Although some studies have evaluated the delay introduced by the virtualization of network functions [5]–[9], little effort has been devoted to the impact to the service availability in the context of NFV. Service chain availability, depends on many factors such as availability of commodity hardware, host operating system, network function software, as well as the links over which the service chains are routed [10]. The placement of the network function in NFV based networks is very flexible thanks to the fact that software instances can be installed at any general purpose hardware with enough available spare capacity. The function placement has a critical impact on the performance guarantees that operator can provide to their customers, as well as on the cost of the service provisioning.

This paper presents two function placement strategies that minimize the service deployment cost for the operator, without compromising the quality of service promised in the SLAs. The optimal solution is found by solving a corresponding Integer Linear Program (ILP). Since the computational complexity, and consequently the execution time, of the ILP becomes impractical for big networks with large number of service requests, we propose an efficient heuristic that is able to find nearly optimal solution in much shorter time. This paper shows that the proposed virtual function placement solutions are able to significantly reduce the risk of SLA violations that an operator would get when considering only delay constraints. Furthermore, we also show that guaranteeing both delay and availability requires only a limited increase of used network resources.

The rest of the paper is organized as follows. Section II provides an overview of the related work in areas of service availability and function placement problem. In sections III and IV, SLA models and problem formulation are presented. The simulation setup and the results are discussed in section V. We conclude the paper with a summary and an outlook of the future work.

II. Related work

In NFV deployments, the placement of network functions has a critical impact on the performance guarantees that operator can provide to their customers, as well as on the cost of the service provisioning. The problems of function

placement and service chain embedding have been recognized as important research challenges [11].

A formal mathematical analysis is provided in [12]. Here, function placement in NFV is described as a combination of two NP-hard problems: the Facility Location Problem (FLP) and the Generalized Assignment Problem (GAP), and as such intractable for large problem instances. Approximation algorithm based on the ILP relaxation and rounding have been also proposed to solve a capacitated NFV location problem.

Several studies focus on a particular use case, such as packet/optical data centers [13], enterprise [14] and mobile core networks [5], [15], [16]. However, with exception of [5], none of the studies considers service chain QoS requirements. In [5] the authors show how delay increases when network functions are fully virtualized and how the delay constrains of the service chain affects the optimal placement of the functions. The authors in [6] show that service chains with virtual network functions, despite having the larger processing delay, can provide lower end to end delay compared to the traditional infrastructures. The increase of the processing delay specific for virtualization of network functions has been studied in [7] and [8]. The increase of the processing delay caused by the side effects of multicore deployment and effects of multiple network functions sharing the same physical hardware are analyzed in [7], while the dependency between the load and the forwarding latency are presented in [8]. The study in [9] show a trade-off between the service chain latency, the number of deployed host nodes in the network and the remaining data rate on network links. However, none of the presented studies has considered service chain availability requirements.

In [17] the authors compare different dedicated protection schemes. The proposed solution provide resiliency against single link or node failures, but require double amount of the resources compared to the unprotected scenario. Our function placement strategy is able to increase the availability of the service chains with much lower network resource usage.

III. System Model and Problem Formulation

We consider network functions that are deployed as software components on general purpose hardware and services that are realized by traversing an ordered set of network functions, referred to as service chains. Multiple instances of the same network function can exist and that one instance can be shared among multiple service chains. These two facts give the operator greater flexibility when setting up the services.

Let us consider as an example, a video conferencing service between two customer premises, which requires the following service chain: Network Address Translation (NAT), firewall (FW) and Intrusion Detection System (IDS), as presented in the Fig. 1. In this example, NAT and FW are collocated at node n_1 and IDS in n_2.

Operators charges for their services based on the performance guarantees specified in SLAs, that typically contain the parameters such as minimum guaranteed bandwidth, maximum end to end delay and service availability. The ultimate goal of the operator is to provide a service to the users with

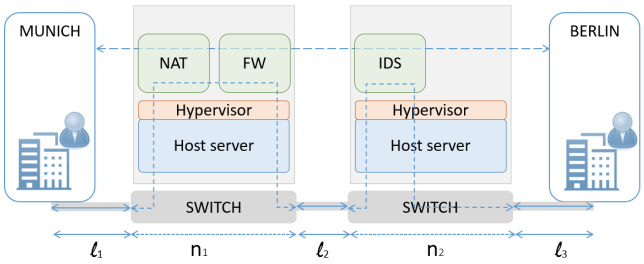


Fig. 1: Service is realized by traversing an ordered set of network functions, in this case Network Address Translation (NAT), firewall (FW) and Intrusion Detection System (IDS).

the QoS guarantees, as specified in their respective SLAs, at the minimum cost (i.e., minimum required resources). Let us present the considered delay and availability service chain (q) models:

- Service chain end to end delay: It can be expressed as the sum of the processing delays of all network nodes and propagation delay of all traversed links.
- Service chain availability: It can be expressed as the product of the availability of all the network functions $n \in q$ and all the traversed links (or segments) $l \in q$. This product can be linearized by applying the $\log$ function, so it can be easily included in the ILP optimization problems.

$$A_q = \prod_{l \in q} A_l . \prod_{n \in q} A_n \Rightarrow \log A_q = \sum_{l \in q} \log A_l + \sum_{n \in q} \log A_n \tag{1}$$

Based on the analysis in [10] we distinguish two types of function faults: (i) faults related to the physical host server (e.g. hardware, host operating system, hypervisor, VM manager), whose availability is denoted as A_n^{Host} and (ii) the faults related to the software instances of virtual network functions (e.g. software bugs, configuration errors), whose availability is denoted as A_{VNFi}.

$$\log A_n = \log A_n^{Host} + \sum_{VNFi \in n} \log A_{VNFi} \tag{2}$$

The study in [18] showed that protection has limited benefits against network functions faults, such as load balancers, since the root cause (e.g. error in the configuration script or a software bug) cannot be mitigated by simply replicating the device. Moreover, in the case of many network functions, like stateful firewall or IDS, the synchronization between the working and the backup replica would be required, which would introduce additional overhead and complexity. Our service provisioning strategy maps service chains to network components to maximize the compliance of their estimated availability with SLA, without relying on any protection scheme.

IV. Optimization of service chain embedding and placement of network functions

An operator's service provisioning strategy should give an answer to i) where the the instances of virtual network functions should be deployed, ii) which service chain has to be mapped to which function instance and iii) how the service chains have to be routed through the network. In the following sections we present two strategies for service chain provisioning and the placement of the virtual network functions that minimize the risk of SLA violation, while minimizing the deployment cost for the operators.

A. Input parameters

1) Physical network substrate: The network topology is defined as a graph $G = (V, E)$ where V is the set of network nodes, and E is the set of communication links. A network node is a switch that can have physical host servers attached to it. A host node is characterized by its physical capacity C_i^P, where P can stand for any physical resource, such as processing power, memory or storage, and their availability A_n. A communication link $(i, j) \in E$ is characterized by its bandwidth B_{ij}, propagation delay of D_{ij}, and availability A_{ij}.

2) Virtual network functions: F denotes a set of all supported virtual network functions. Each virtual network function $v \in F$ consumes certain amount of physical resources C_v^P and can handle a limited amount of traffic B_v. The function introduces processing delay D_v and has an availability A_v.

3) Service function chains: S denotes a set of all service chains that have to be provisioned in the network. A service chain q consists of nodes, that can be physical endpoints $S_q^P \subseteq V$ or a virtual network functions $S_q^V \subseteq F$, and virtual links between them S_q^L. A service chain request is characterized by a data rate B_q, maximum allowed end-to-end delay D_q and minimum availability A_q.

B. QoS-aware function placement an ILP optimization problem (q-ILP)

Let us define the following binary variables:

- $x_{i,v}$ indicates if a virtual network function v is mapped to a physical node i
- $y_{i,q,m}$ indicates if a node $m \in S_q^P \cup S_q^V$ of the chain q is mapped to a physical node i
- $z_{ij,q,kl}$ indicates if a virtual link (k, l) of a chain q is mapped to a physical link (i, j)

We also define auxiliary binary variables h_i to indicate if a physical host node i is used to host any virtual function, and $h_{i,q}$ to indicate if node i is used by the service chain q.

The problem objective is to minimize the cost of the resources that have a direct impact on capital and operational expenditures for the network operator while coping with all QoS constraints (delay and availability). The objective function can be expressed as:

$$\min\ c_h \sum_{i \in V} h_i + c_v \sum_{i \in V} \sum_{v \in F} x_{i,v} + c_l \sum_{ij \in E} \sum_{q \in S} \sum_{kl \in S_q^L} z_{ij,q,kl} B_q$$

where the first term considers the host server costs (including the site opening and equipment installation cost) which is multiplied by the number of servers needed to host virtual network functions; the second term includes the cost associated to the network function licenses, which is proportional to the number of network function instances; and the last term expresses the link cost which is related to the used bandwidth per link (important when leasing capacity). The relative importance of each cost (host c_h, network function licenses c_v and link transit cost c_l) depend on the operator's cost model and should reflect the market price of each of the resources. They are given as an input parameter.

The placement constraints can be grouped into several categories.

1) Capacity constraints: The resources consumed by the virtual network functions hosted by a server i cannot exceed the available physical server capacity.

$$\sum_{v \in F} x_{i,v} C_v^P \leq C_i^P; \forall i \in V \tag{3}$$

The traffic handling capacity of virtual network functions must be enough to support all service function chains mapped to it.

$$\sum_{q \in S;\ \text{if } v \in S_q^V} y_{i,q,v} B_q \leq x_{i,v} B_v; \forall i \in V; \forall v \in F \tag{4}$$

The link bandwidth should be larger or equal to the capacity required by all the service chains using that link.

$$\sum_{q \in S} \sum_{kl \in S_q^L} z_{ij,q,kl} B_q \leq B_{ij}; \forall (i, j) \in E; \tag{5}$$

2) Placement constraints: Physical endpoints of the chain S_q^P have to be mapped to the corresponding nodes in the physical substrate.

$$y_{i,q,i} = 1; \forall i \in V; \forall q \in S; \forall i \in S_q^P \tag{6}$$

$$y_{i,q,j} = 0; \forall i \in V; \forall q \in S; \forall j \in S_q^P; \text{if } i \neq j \tag{7}$$

If a virtual network function $v \in S_q^V$ is mapped to a physical node i, there has to be a function of a requested type available in that node.

$$y_{i,q,v} \leq x_{i,v}; \forall i \in V; \forall q \in S; \forall v \in S_q^V \tag{8}$$

One virtual network function $v \in S_q^V$ of a chain q can be mapped to only one physical node in the network.

$$\sum_{i \in V} y_{i,q,n} = 1; \forall q \in S; \forall v \in S_q^V \tag{9}$$

Additionally we need to include the host mapping indicator variables h_i and $h_{i,q}$.

$$h_i = \begin{cases} 1, \text{if } \sum_{v \in F} x_{i,v} > 0 \\ 0, \text{otherwise} \end{cases} \tag{10}$$

$$h_{i,q} = \begin{cases} 1, \text{if } \sum_{v \in F} y_{i,q,v} > 0 \\ 0, \text{otherwise} \end{cases} \tag{11}$$

3) Routing constraints: The flow conservation law has to hold for mapping of virtual links (k,l) of chain q to the physical links (i,j).

$$\sum_{ij\in E} z_{ij,q,kl} - \sum_{ji\in E} z_{ji,q,kl} = y_{i,q,k} - y_{i,q,l}; \quad \forall i \in V; \forall q \in S; \forall (k,l) \in S_q^L \tag{12}$$

Loops can be prevented by allowing at most one outgoing and at most one incoming edge per node i being assigned to a chain q. This applies only to linear service chains.

$$\sum_{ij\in E} \sum_{kl\in S_q^L} z_{ij,q,kl} \leq 1; \forall i \in V; \forall q \in S; \tag{13}$$

$$\sum_{ij\in E} \sum_{kl\in S_q^L} z_{ij,q,kl} \leq 1; \forall j \in V; \forall q \in S; \tag{14}$$

4) QoS constraints: Maximum end-to-end delay and minimum availability of the service chain have to be guaranteed.

$$\sum_{i\in V} \sum_{v\in S_q^V} y_{i,q,v} D_v + \sum_{ij\in E} \sum_{kl\in S_q^L} z_{ij,q,kl} D_{ij} \leq D_q \tag{15}$$

$$\sum_{i\in V} h_{i,q} \log(A_i) + \sum_{i\in V} \sum_{v\in S_q^V} y_{i,q,v} \log(A_v) + \\ + \sum_{ij\in E} \sum_{kl\in S_q^L} z_{ij,q,kl} \log(A_{ij}) \geq \log(A_q) \tag{16}$$

C. QoS-aware Service Chain Embedding and function Placement (q-SCP)

We propose a solution based on greedy heuristic, *q-SCEP*, that is able to find near optimal solution in much shorter time by sequentially embedding the service chains. The chains are sorted in a decreasing order based on the estimated cost of their embedding and difficulty of the compliance with the QoS guarantees that are specified in SLA. The minimum cost is estimated as a weighted sum of the shortest QoS-constrained path, number of the network functions in the chain and the minimum number of the servers that are needed to host them. For every service chain the algorithm first finds the shortest QoS-constrained path between the physical endpoints of the service chain. The path is then extended to include the network functions specified by the chain. The outline of the overall procedure is illustrated in the Alg. 1.

For every network function in the chain a set of candidate nodes are evaluated. The best candidate is the one that induces the lowest additional cost. In the first step, the set of candidate nodes where the function v is already deployed is evaluated. The cost of selecting those candidates induces the additional link transit cost due to the path extension. If the additional cost is higher than the cost of deployment of the new instance of network function, the set of candidates with enough spare capacity is also considered. The cost of selecting these candidates includes the cost of installing an additional software license, as well as the cost of the path extension. If the cost of the best candidate at this point exceeds the cost of the installation of the new hardware, the third set of candidates is considered. These candidates induce the additional cost of new hardware and new software license, and in some cases the path extension cost if there is no space to install the hardware along the shortest path. If two candidates lead to the same additional cost, the one with the highest betweenness centrality is selected. The procedure of selecting the best candidate for the virtual function is summarized in Alg. 2.

Algorithm 1 QoS-aware Service Chain Embedding and virtual function Placement (q-SCEP)

Input: Physical network substrate (G), models of virtual network functions (F), set of requested service chains (S) and operator's cost model (c_h, c_v, c_l)
Output: Placement of servers (H), function placement (X), mapping to chains (Y) and routing of service chains (Z)

```
SortedScRequests = sort the service chains based on the minimum embedding cost in descending order
for q ∈ SortedScRequests do
    minPath = minQosConstraintedPath()
    minCost = cost(minPath)
    for vnf ∈ S_q^V do
        vPlacement = bestCandidate(s, t, vnf, qos)
        Update minPath, currentCost, QoS budget
        Update residual capacity
    end for
end for
return H, X, Y, Z
```

In order to estimate the minimum path stretching cost, we have to find least cost path satisfying the QoS constraints (bandwidth, delay and availability) specified in the service chain SLA. The edges without enough spare capacity are removed from the graph during search. The cost of the edge is a weighted sum of the hop count (link transit cost), propagation delay and logarithm of its availability. Delay and availability cost factors are scaled to reflect the overall contribution to the QoS budget, and to represent non negative values (less than one for the feasible paths).

$$\sum_{ij\in path} D_{ij}/D_{budget} = D_{path}/D_{budget}$$

$$\sum_{ij\in path} \log A_{ij}^{-1} / \log A_{budget}^{-1} = \log A_{path}^{-1} / \log A_{budget}^{-1}$$

Initially, the highest weight is assigned to the hop count (γ), and small weights ($\epsilon \ll 1$) to the availability and delay of the edge, and the shortest path is found. In this way, if more than one path with the smallest hop count is found, the one contributing least to the QoS budget is selected. If the shortest path found in this way does not satisfy the QoS constraint, the weights of the links are updated proportional to the QoS violation. After several iterations the highest weight will be given to the QoS constraint that is the most

Algorithm 2 Best Candidate

Input: G, s, t, vnf, QoS $(B, D_{budget}, A_{budget})$
Output: vnf placement that induces the least additional cost

$bestCandidate = None$, $bestCost = \infty$
for $n \in vnfCandidates$ **do**
  $minPath = minQosConstraintedPath(s, t, n, qos)$
  $minCost = c_l.B.cost(minPath)$
  Update $bestCandidate$
end for
if $bestCost - currentCost \geq c_v$ **then**
  for $n \in hostCandidates$ **do**
    $minPath = minQosConstraintedPath(s, t, n, qos)$
    $minCost = c_l.B.cost(minPath) + c_v$
    Update $bestCandidate$
  end for
end if
if $bestCost - currentCost \geq c_v + c_h$ **then**
  for $n \in newCandidates$ **do**
    $minPath = minQosConstraintedPath(s, t, n, qos)$
    $minCost = c_l.B.cost(minPath) + c_v + c_h$
    Update $bestCandidate$
  end for
end if
return $bestCandidate$

Algorithm 3 minQosConstrainedPath

Input: G, s, t, n, QoS $(B, D_{budget}, A_{budget})$
Output: Shortest path between s and t, containing intermediate node n, and satisfying the QoS constraints

Initialisation :
$\alpha = \beta = \epsilon, \gamma = 1 - 2\epsilon$
$\omega_{ij} = \alpha D_{ij}/D_{budget} + \beta \log A_{ij}^{-1} / \log A_{budget}^{-1} + \gamma$
for $attempt \leq maxAttempts$ **do**
  $path_1 = shortestPath(G, s, n, weight)$
  $path_2 = shortestPath(G, n, t, weight)$
  $path = path_1 + path_2$
  if $D(path) < D_{budget}$**and**$A(path) > A_{budget}$ **then**
    return $path$
  end if
  if $D(path) > D_{budget}$ **then**
    Update α proportional to QoS violation
  end if
  if $A(path) < A_{budget}$ **then**
    Update β proportional to QoS violation
  end if
  Update weights as:
  $\omega_{ij} = \alpha D_{ij}/D_{budget} + \beta \log A_{ij}^{-1} / \log A_{budget}^{-1} + \gamma$
end for
return $False$

difficult to satisfy. Note that if, for an example, the weights $(\alpha, \beta, \gamma) = (1, 0, 0)$ still do not lead to the path with delay lower than the one specified in the budget, the feasible path does not exist. Maximum number of iterations can be set in advance to limit the execution time. Here we limit it to $R_{iter} = 10$. This algorithm gives a good approximation for shortest QoS constrained path, for all the studied scenarios. The pseudo code for this subroutine is presented in Alg. 3.

D. Baseline algorithm

We also present a baseline algorithm to compare the effectiveness of our proposed heuristic. A feasible solution of the service chain embedding and function placement problem can be found simply by deploying the virtual network functions along the shortest QoS-constrained paths for every service chain. The number of installed host servers and network function licenses can be reduced by reusing the hosts and functions already available along the shortest path. This approach is quite fast, as it computes $|S| * R_{iter}$ shortest paths, and gives a good upper bound for comparing the cost efficiency of the proposed heuristic.

V. Evaluation

The proposed heuristic was compared against optimal solution obtained by solving the ILP optimization problem and the baseline approach based on the shortest paths. The criteria used to evaluate the goodness of the heuristic were its ability to fulfil the SLA requirements, cost efficiency and solving time.

A. Experimental setup

Optimal solution was found by solving the ILP optimization problem with Gurobi [19] solver on Intel® Core™i7-4790 @3.60 GHz machine with 16 GB RAM memory with Ubuntu 14.04 LTS operating system.

The network used in the simulations was "nobel-germany" available in SNDlib database [20]. The hardware for hosting the servers can be installed at any node in the network. Host servers have the capacity to host up to 8 virtual network functions, and have the availability of 99.9%. The capacity of the physical links was set to 1 Gbps.

Models of virtual network functions and service mix used in the simulations were based on the study in [7]. Considered functions were Network Address Translation (NAT), firewall (FW), traffic monitor (TM), WAN Optimization Controller (WOC), Video Optimization Controller (VOC) and Intrusion Detection System (IDS). All functions were assumed to have the same traffic handling capacity (200 Mbps) and introduce the same processing delay (0.5 ms). Assumed availability of the network function's software was 99.9%.

We have chosen the video conferencing and online gaming as representative examples of the services sensitive to connection interruptions. The QoS parameters related to each chain are specified in the Table I. Assumed availability requirements were 99%. Several users requesting the same service type (e.g. video conferencing or online gaminig) between random source and destination points represent one service request. In every experiment between one and 40 service simultaneous requests are embedded. We study a heterogeneous service mix, where

service requests are equally split between two service types.

TABLE I: Service chain model parameters [7]

Service	SFC	Data rate	Max delay
Video	NAT-FW-TM-VOC-IDS	4 Mbps	100 ms
Gaming	NAT-FW-VOC-WOC-IDS	50 Kbps	60 ms

The relative importance between the cost of installation of host server hardware, the cost of installation of the network function software license and the of allocating 1 Mbps of bandwidth over one link is 100:10:1.

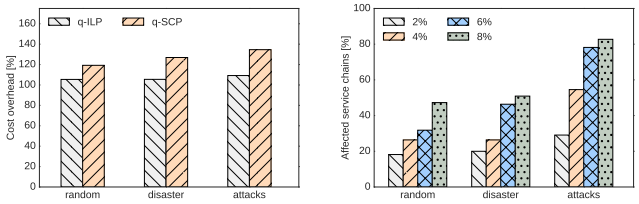
Due to the space limitations we present only the most relevant results, and omit the sensitivity study of the cost efficiency to the stated input parameters.

B. SLA fulfilment

We test the performance of our service provision strategies for different distributions of link unavailabilities, and show the consequence of optimizing only the cost, without taking the service availability into account.

Nominal unavailability of the links is proportional to their length, which reflects the probability of the cable cuts. We assume that the link unavailability is as 0.02% per 100 km, which is a realistic assumption for buried optical cables. Additionally, several links are expected to fail with higher probability, than the others, because of their age, a natural disaster in a particular region of the network or an intentional attack. We simulate this by increasing the unavailability of the high risk link 50 times their nominal value. In the first scenario high risk links are selected randomly, in the second by proximity to the epicenter of the disaster and in the third by their betweenness centrality. In every scenario between 2% and 8% of the links are affected.

In Fig.2 the cost of availability awareness in the three scenarios is presented. The cost of service provisioning of our availability-aware schemes are compared w.r.t. cost optimal strategy that does not take into account service availability. It can be seen in the Fig. 2a that the cost of the service provisioning with q-ILP is less than 20% in all observed scenarios. The cost overhead is slightly higher for the q-SCP scheme, and it is highest in the attack scenario, around 40%.



(a) Cost of availability awareness (b) Risk of SLA violation

Fig. 2: Cost and the benefits of service availability awareness.

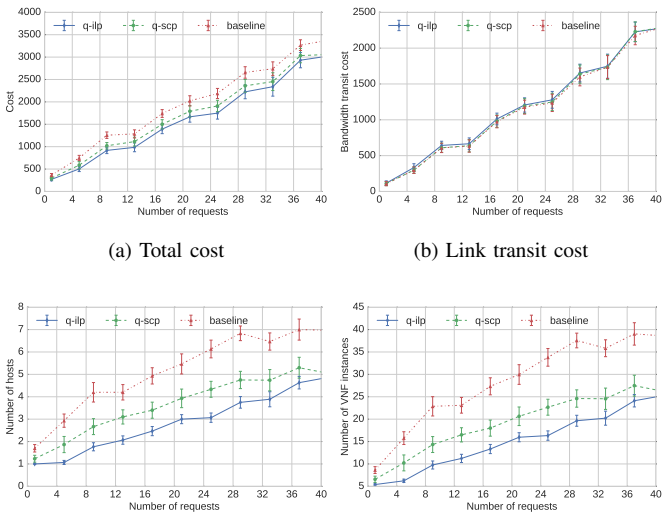
However, without taking the availability into account, service SLA may be violated. In the Fig.2 the percentage of service request whose SLA was not fulfilled is presented.

C. Cost efficiency

Next we compare the cost efficiency of the proposed heuristic for different network planning parameters, relative cost of the resources, size of the data centers and expected service mix.

Cost for different number of service requests is presented in the Fig.3a. It can be seen that the q-SCP heuristic was very close to the optimal solution obtained by solving q-ILP, less than 5% difference for the observed scenario, much closer than the baseline scheme based on the shortest paths.

Link transit cost, shown in Fig.3d, was almost the same as in baseline scenario based on the shortest paths. Fig.3b and Fig.3c show the number of opened sites and the number of network function software licenses respectively. Both cost components follow the same trend and increase with the number of service requests.



(a) Total cost (b) Link transit cost

(c) Number of host servers (d) Network function instances

Fig. 3: Cost of the service provisioning in NFV deployments. The cost is composed of the link transit cost, the cost of the hosts and network function software instances.

D. Solving time

We compared the solving time for two German networks, 'nobel-germany' and 'germany50' [20], for different number of the service requests. In can be seen in the Table II that the solving time of the q-ILP is significantly higher than the solving time of the heuristic, 4.190 s compared to 0.581 s for $|S| = 20$ chains for network 'germany50'. Solving time of the q-ILP problem grows exponentially with the size of the problem, while solving time grows almost linearly with the number of service requests. This is due to the fact ther q-SCP computes in the worst case $|S|.R_{iter}.|V|.|S_q^V|$ shortest paths (where $|S_q^V|$ is the number of network functions in the chain), compared to $|S|.R_{iter}$ in the baseline scenario. If Dijkstra is used for the shortest path computation, the worst case runtime (of shortest path computation) is $O(|E| + |V| \log |V|)$.

TABLE II: Solving times [s] for two German topologies when $|S| = 20$ service chains are embedded in the network

Network	Nodes	Edges	q-ILP	q-SCP	Baseline
nobel-germany	17	51	2.424	0.156	0.058
germany50	50	88	4.190	0.581	0.179

VI. Conclusion and future work

In this paper we study the problem of delay and availability based service differentiation in NFV networks. Since availability was not considered so far in NFV placement problem, we provide a comprehensive model of service chain availability, and use it to solve the problem of cost efficient service provisioning. We designed two mechanisms to place virtualized network functions so that support service differentiation in terms of delay and availability while minimizing the associated costs. The proposed strategies, one based on ILP and one based on the heuristic, are compared in terms of their cost efficiency in different scenarios and their computational complexity. The proposed heuristic was able to adapt to different availability distributions and find the function placement solution without any SLA violation, while maintaining the high cost efficiency. In the particular case study, we show that availability and delay guarantees can be provided with less than 20% increase of the cost.

In future we plan to extend our function placement schemes to the online scenario, when service request demands are not known in advance.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671648 (VirtuWind).

References

[1] ETSI GS NFV 001 V1.1.1, "Network Functions Virtualisation (NFV); Use Cases," 2013.

[2] P. Quinn and T. Nadeau, "RFC 7948, Problem Statement for Service Function Chaining," *Internet Engineering Task Force (IETF), ed*, 2015.

[3] A. Mykkeltveit and B. E. Helvik, "Adaptive management of connections to meet availability guarantees in slas," in *2009 IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2009, pp. 545–552.

[4] M. Durvy, C. Diot, N. Taft, and P. Thiran, "Network availability based service differentiation," in *International Workshop on Quality of Service*. Springer, 2003, pp. 305–325.

[5] 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, & challenges*. ACM, 2014, pp. 33–38.

[6] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 98–106.

[7] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*. IEEE, 2015, pp. 191–197.

[8] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015, pp. 171–177.

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

[10] D. Cotroneo, L. De Simone, A. Iannillo, A. Lanzaro, R. Natella, J. Fan, and W. Ping, "Network function virtualization: challenges and directions for reliability assurance," in *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 37–42.

[11] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[12] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[13] 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, pp. 1565–1570, 2015.

[14] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE, 2014, pp. 418–423.

[15] H. Ko, G. Lee, I. Jang, and S. Pack, "Optimal middlebox function placement in virtualized evolved packet core systems," in *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. IEEE, 2015, pp. 511–514.

[16] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2014, pp. 2402–2407.

[17] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Virtual network function placement for resilient service chain provisioning," in *Reliable Networks Design and Modeling (RNDM), 2016 8th International Workshop on*. IEEE, 2016.

[18] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 350–361.

[19] G. Optimization, "Inc.,gurobi optimizer reference manual, 2014," 2014.

[20] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0survivable network design library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.