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, offering much greater flexibility and lower cost of the service deployment for the network operators. At the same time, replacing the network functions implemented in purpose built hardware with software modules poses a great challenge for the operator to maintain the same level of performance. The grade of service promised to the end users is formalized in the Service Level Agreement (SLA) that typically contains the QoS parameters, such as minimum guaranteed data rate, maximum end to end latency, port availability and packet loss. State of the art solutions can guarantee only data rate and latency requirements, while service availability, which is an important service differentiator is mostly neglected. This paper focuses on the placement of virtualized network functions, aiming to support service differentiation between the users, while minimizing the associated service deployment cost for the operator. Two QoS-aware placement strategies are presented, an optimal solution based on the Integer Linear Programming (ILP) problem formulation and an efficient heuristic to obtain near optimal solution.**

Considering a national core network case study, we show the cost overhead of availability-awareness, as well as the risk of SLA violation when availability constraint is neglected. We also compare the proposed function placement heuristic to the optimal solution in terms of cost efficiency and execution time, and demonstrate that it can provide a good estimation of the deployment cost in much shorter time.

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]. Service chaining with virtualized network functions offer greater flexibility in the service provisioning and lower required resources for the network operators. First studies of such benefits have been demonstrated in various use cases ranging from mobile and fixed access to Content Delivery Networks (CDN) [1].

Network operators offer a wide service portfolio to their end customers. The details of a service grade promised 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]. Several studies have already focused on QoS-awareness in terms of data rate and latency [5]–[9], while little effort has been devoted to the service availability. Service availability in NFV-based networks, 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 virtual network functions 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.

Considering a national core network case study, we show that the proposed QoS-aware placement strategies are able to provide delay and availability guarantees in a cost-efficient manner, and that the risk of SLA violation is considerably high when service availability constraint is neglected. We also compared accuracy of the proposed heuristic to the optimal solution, and demonstrate that it is able to provide good estimate of the service deployment cost. Finally, we show that the execution time of the proposed heuristic scales well with the size of the problem.

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]. A placement of virtual network functions 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 proposed. Other studies focused 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 considered 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 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 without providing the explicit guarantees for the service availability. 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

Network functions are deployed as software components running in virtual machines on general purpose hardware, while 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 in the network. Hardware resources can be shared between different functions and functions can be shared among multiple service chains. An example of video conferencing service between two customer premises, which requires the following service chain: Network Address Translation (NAT), firewall (FW), Video Optimization Controller (VOC) and Intrusion Detection System (IDS), is illustrated in the Fig. 1. In this example, NAT and FW are collocated at node n_1 and VOC and IDS in n_2.

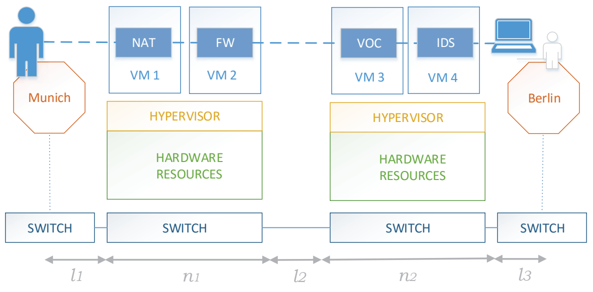


Fig. 1: Service is realized by traversing an ordered set of network functions, in this case Network Address Translation (NAT), firewall (FW), Video Optimization Controller (VOC) and Intrusion Detection System (IDS).

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. Delay of the service chain (q) can be expressed as the sum of propagation delay of all traversed links $l \in q$ and processing delays of all network nodes $n \in q$.

$$D_S = \sum_{l \in S} D_l + \sum_{n \in S} D_n$$

Service chain availability 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.

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

Based on the analysis in [10] we distinguish two types of network function failures: (i) failures 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 failures related to the software instances of virtual network functions (e.g. software bugs, configuration errors), whose availability is denoted as A_{VNF_i}.

$$\log A_n = \log A_n^{Host} + \sum_{\mathrm{VNF}_i \in n} \log A_{\mathrm{VNF}_i}$$

The study in [18] showed that protection has limited benefits in the case of the failures of complex network function, 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 the stateful network functions, such as 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 strategies map service chains to network components in a way that ensures the compliance of their estimated availability with service availability specified in the SLA, without relying on any protection scheme.

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

In the following sections we present two QoS-aware function placement strategies that minimize the service deployment cost for the operators.

A. Input parameters

1) Physical network substrate: The network topology is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of network nodes, and $\mathcal{E}$ is the set of communication links. A network node is represents a forwarding device, a switch or a router, 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 its availability A_n. A communication link $(i, j) \in \mathcal{E}$ is characterized by its bandwidth B_{ij}, propagation delay of D_{ij}, and availability A_{ij}.

2) Virtual network functions: $\mathcal{F}$ represents a set of all supported virtual network functions. Each virtual network function $v \in \mathcal{F}$ 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: $\mathcal{S}$ 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 as 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 (bandwidth, delay and availability). The objective function can be expressed as:

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

where the first term considers the host server costs (including the site opening and equipment installation cost); the second term includes the cost associated to the network function licenses; 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 component (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.

The placement constraints are 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.

$$\sum_{v\in\mathcal{F}} x_{i,v} C_v^P \le C_i^P; \quad \forall i \in \mathcal{V} \tag{1}$$

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

$$\sum_{q\in\mathcal{S};\ \text{if}\ v\in S_q^V} y_{i,q,v} B_q \le x_{i,v} B_v; \quad \forall i \in \mathcal{V}; \forall v \in \mathcal{F} \tag{2}$$

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

$$\sum_{q\in\mathcal{S}} \sum_{kl\in S_q^L} z_{ij,q,kl} B_q \le B_{ij}; \quad \forall (i,j) \in \mathcal{E} \tag{3}$$

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

$$y_{i,q,i} = 1; \quad \forall i \in \mathcal{V}; \forall q \in \mathcal{S}; \forall i \in S_q^P \tag{4}$$

$$y_{i,q,j} = 0; \quad \forall i \in \mathcal{V}; \forall q \in \mathcal{S}; \forall j \in S_q^P; \text{if } i \ne j \tag{5}$$

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.

$$y_{i,q,v} \le x_{i,v}; \quad \forall i \in \mathcal{V}; \forall q \in \mathcal{S}; \forall v \in S_q^V \tag{6}$$

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

$$\sum_{i\in\mathcal{V}} y_{i,q,n} = 1; \quad \forall q \in \mathcal{S}; \forall v \in S_q^V \tag{7}$$

Host mapping indicator variables h_i and $h_{i,q}$ are defined as:

$$h_i = \begin{cases} 1, \text{if } \sum_{v\in F} x_{i,v} > 0 \\ 0, \text{otherwise} \end{cases} \tag{8}$$

$$h_{i,q} = \begin{cases} 1, \text{if } \sum_{v\in F} y_{i,q,v} > 0 \\ 0, \text{otherwise} \end{cases} \tag{9}$$

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).

$$\sum_{ij\in\mathcal{E}} z_{ij,q,kl} - \sum_{ji\in\mathcal{E}} z_{ji,q,kl} = y_{i,q,k} - y_{i,q,l}; \quad \forall i \in \mathcal{V}; \forall q \in \mathcal{S}; \forall (k,l) \in S_q^L \tag{10}$$

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.

$$\sum_{ij\in\mathcal{E}} \sum_{kl\in S_q^L} z_{ij,q,kl} \leq 1; \quad \forall i \in \mathcal{V}; \forall q \in \mathcal{S}; \tag{11}$$

$$\sum_{ij\in\mathcal{E}} \sum_{kl\in S_q^L} z_{ij,q,kl} \leq 1; \quad \forall j \in \mathcal{V}; \forall q \in \mathcal{S}; \tag{12}$$

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

$$\sum_{i\in\mathcal{V}} \sum_{v\in S_q^V} y_{i,q,v} D_v + \sum_{ij\in\mathcal{E}} \sum_{kl\in S_q^L} z_{ij,q,kl} D_{ij} \leq D_q \tag{13}$$

$$\sum_{i\in\mathcal{V}} h_{i,q} \log(A_i) + \sum_{i\in\mathcal{V}} \sum_{v\in S_q^V} y_{i,q,v} \log(A_v) + + \sum_{ij\in\mathcal{E}} \sum_{kl\in S_q^L} z_{ij,q,kl} \log(A_{ij}) \geq \log(A_q) \tag{14}$$

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

We propose a solution based on greedy heuristic, *q-SCP*, 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. 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 vnf is already deployed, is evaluated. The cost of selecting those candidates incurs only 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 host 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 idea behind this is that the candidate with higher betweenness centrality is more likely to lay in the shortest path of some of the future requests. 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-SCP)

```
Input: Physical network substrate ($\mathcal{G}$), models of virtual network functions ($\mathcal{F}$), set of requested service chains ($\mathcal{S}$) 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 ($Z$)
1:  $SortedScRequests$ = sort the service chains based on the minimum embedding cost in descending order
2:  for $q \in SortedScRequests$ do
3:      Start from physical endpoints $s, t \in S_q^P$
4:      $minPath = minQosConstraintedPath(s, t, qos)$
5:      $minCost = cost(minPath)$
6:      for $vnf \in S_q^V$ do
7:          $vPlacement = bestCandidate(s, t, vnf, qos)$
8:          Update $minPath$, $currentCost$, QoS budget
9:          Update residual capacity
10:     end for
11: end for
12: 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

Algorithm 2 Best Candidate

```
Input: $\mathcal{G}, s, t, vnf$, QoS $(B, D_{budget}, A_{budget})$, $currentCost$
Output: $vnf$ placement that induces the least additional cost
1:  $bestCandidate = None$, $bestCost = \infty$
2:  for $n \in vnfCandidates$ do
3:      $minPath = minQosConstraintedPath(s, t, n, qos)$
4:      $minCost = currentCost + c_l.B.cost(minPath)$
5:      Update $bestCandidate$
6:  end for
7:  if $bestCost - currentCost \geq c_v$ then
8:      for $n \in hostCandidates$ do
9:          $minPath = minQosConstraintedPath(s, t, n, qos)$
10:         $minCost = currentCost +$
                $c_l.B.cost(minPath) + c_v$
11:         Update $bestCandidate$
12:     end for
13: end if
14: if $bestCost - currentCost \geq c_v + c_h$ then
15:     for $n \in newCandidates$ do
16:         $minPath = minQosConstraintedPath(s, t, n, qos)$
17:         $minCost = currentCost +$
                $c_l.B.cost(minPath) + c_v + c_h$
18:         Update $bestCandidate$
19:     end for
20: end if
21: return $bestCandidate$
```

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 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, then 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 $R_{iter} * |S|$ shortest paths, and gives a good upper bound for comparing the cost efficiency of the proposed heuristic.

Algorithm 3 minQosConstrainedPath

```
Input: $\mathcal{G}, s, t, n$, QoS $(B, D_{budget}, A_{budget})$
Output: Shortest path between $s$ and $t$, containing intermediate node $n$, and satisfying the QoS constraints
    Initialisation :
    $\alpha = \beta = \epsilon, \gamma = 1 - 2\epsilon$
    $\omega_{ij} = \alpha D_{ij}/D_{budget} + \beta \log A_{ij}^{-1} / \log A_{budget}^{-1} + \gamma$
1:  for $attempt \leq maxAttempts$ do
2:      $path_1 = shortestPath(s, n, weight)$
3:      $path_2 = shortestPath(n, t, weight)$
4:      $path = path_1 + path_2$
5:      if $D(path) < D_{budget}$ and $A(path) > A_{budget}$ then
6:          return $path$
7:      end if
8:      if $D(path) > D_{budget}$ then
9:          Update $\alpha$ proportional to QoS violation
10:     end if
11:     if $A(path) < A_{budget}$ then
12:         Update $\beta$ proportional to QoS violation
13:     end if
14:     Update weights as:
15:     $\omega_{ij} = \alpha D_{ij}/D_{budget} + \beta \log A_{ij}^{-1} / \log A_{budget}^{-1} + \gamma$
16: end for
17: return $None$
```

V. Evaluation

The presented QoS-aware function placement strategies, q-ILP and q-SCP, were compared in terms of their cost-efficiency and solving time. Due to the space limitation we omit the sensitivity analysis, and present only the most relevant results of the case study on the national core network.

A. Case study

The network used in the simulations was "nobel-germany" available in SNDlib database [19]. The servers can be installed anywhere in the network and they have the capacity to host up to 8 virtual network functions. The availability of the server hosts with virtualization layer was assumed to be 99.9% The capacity of the physical links was assumed to be 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%.

Video conferencing and online gaming were chosen as representative examples of 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 gaming) 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	Service chain	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.

Optimal solution was found by solving the ILP optimization problem with Gurobi [20] solver running on Intel® Core^{TM}i7-4790 @3.60 GHz machine with 16 GB RAM memory with Ubuntu 14.04 LTS operating system.

B. SLA fulfilment

First, we show the cost of availability-awareness in our placement strategies, and the risk of the SLA violation when service availability constraint is ignored. In order to demonstrate the importance of service availability we consider different network availability profiles.

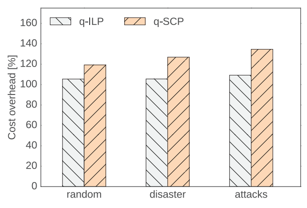
Nominal unavailability of the links is proportional to their length, which reflects the probability of the cable cuts. We assume that the link unavailability is as 0.02% per 100 km, which is a realistic assumption for buried optical cables [21]. Additionally, several links are expected to fail with higher 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.2a 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 (Eq.14). It can be seen in that the cost of the service provisioning with q-ILP is less than 10% in all observed scenarios. The cost overhead is slightly higher for the q-SCP scheme, and it is highest in the attack scenario, around 32%.

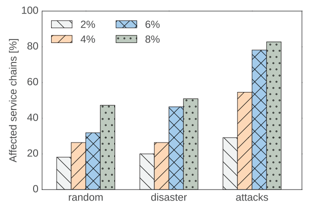
However, without taking the availability into account, service SLA may be violated. In the Fig.2b the percentage of service request whose SLA was not fulfilled is presented. The risk of SLA violation increases with the number of affected links. In the case of attack on the 8% of the links with the highest betweenness centrality, SLA of up to 80% of the service requests may be violated.

C. Cost efficiency

Next, we compare the cost efficiency of the proposed q-SCP heuristic, w.r.t. the optimal q-ILP placement strategy and



(a) Cost of availability-awareness



(b) Risk of SLA violation

Fig. 2: Cost overhead of availability-awareness of the proposed function placement strategies (a) and the risk of SLA violation when service availability constraint is ignored (b).

the baseline algorithm for different number of service requests Fig.3. It can be observed that the q-SCP heuristic was very close to the optimal solution obtained by solving q-ILP, much closer than the baseline scheme based on the shortest paths.

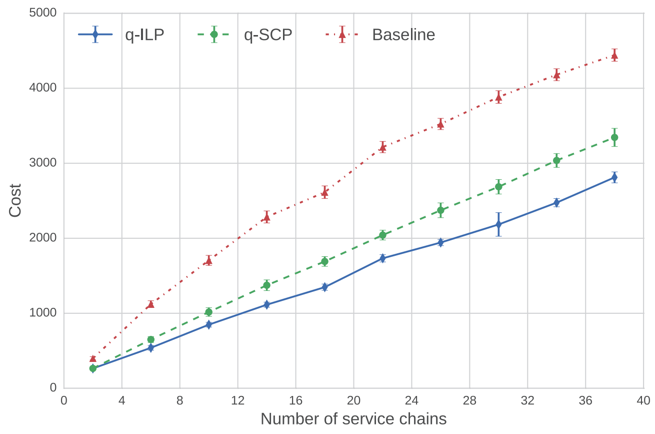


Fig. 3: Cost of the service provisioning in NFV deployments. The total deployment cost is composed of the cost of the hosts, network function software licences and the link transit cost.

D. Solving time

We compared the solving time for two German networks, 'nobel-germany' and 'germany50' [19], 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 $R_{iter}|S||V||S_q^V|$ shortest paths (where $|S_q^V|$ is the number of network functions in the chain), compared to $R_{iter}|S|$ in the baseline scenario. If Dijkstra is used for the shortest path computation, the worst case runtime (of shortest path computation) is $O(|E| + |V| \log |V|)$.

VI. Conclusion and future work

In this paper the problem of QoS-aware function placement in NFV deployments was studied. Service availability is an

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

important QoS differentiator, that has not been considered, so far, in the context of NFV networks. We provided a comprehensive model for service chain availability that accounts for all important factors that contribute to it, such as availability of the host servers, virtualization layer, network function software, as well as the availability of the links that carry the service chain traffic. Based on this model, we design two function placement strategies that support QoS differentiation between users and services, while minimising the cost of the service deployment for the network operator. The first strategy provides an optimal solution by solving a corresponding ILP problem. The second placement strategy was based on the greedy heuristic, and was able to provide close to optimal solution for all studied scenarios in much shorter time. We demonstrated the importance of availability-awareness in the proposed placement strategies. We have shown that availability guarantees can be provided with a little extra cost, compared to the placement strategies that only consider service latency and data rate. On another hand, the risk of SLA violation when service availability constraint is ignored is considerably high, and can affect up to 80% of the service chains for a given case study.

In the future we plan to extend out placement strategies to the online scenario, when service requests are not known in advance and additional factors, such as energy saving, have to be taken into account.

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

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

[21] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, and M. Jaeger, "General availability model for multilayer transport networks," in *DRCN 2005). Proceedings. 5th International Workshop on Design of Reliable Communication Networks, 2005*. IEEE, 2005, pp. 8–pp.