David Dietrich, Ahmed Abujoda, Amr Rizk, Panagiotis Papadimitriou:
Multi-Provider Service Chain Embedding With Nestor. In: IEEE Trans. Network and Service Management, vol. 14, no. 1, p. 91-105 , January 2017.

Multi-Provider Service Chain Embedding with Nestor

David Dietrich[§] Ahmed Abujoda[§] Amr Rizk[†] Panagiotis Papadimitriou[‡]

[§]Institute of Communications Technology, Leibniz Universität Hannover, Germany
{*david.dietrich, ahmed.abujoda*}*@ikt.uni-hannover.de*
[†]Technische Universität Darmstadt, Germany
amr.rizk@kom.tu-darmstadt.de
[‡]Department of Applied Informatics, University of Macedonia, Greece
papadimitriou@uom.gr

***Abstract*—Network Function Virtualization (NFV) decouples network functions (NF) from the underlying middlebox hardware and promotes their deployment on virtualized network infrastructures. This essentially paves the way for the migration of NFs into clouds (*i.e.,* NF-as-a-Service), achieving a drastic reduction of middlebox investment and operational costs for enterprises. In this context, service chains (expressing middlebox policies in the enterprise network) should be mapped onto datacenter networks, ensuring correctness, resource efficiency as well as compliance with the provider's policy. The network service embedding (NSE) problem is further exacerbated by two challenging aspects: (i) traffic scaling caused by certain NFs (*e.g.,* caches, WAN optimizers) and (ii) NF location dependencies. Traffic scaling requires resource reservations different from the ones specified in the service chain, whereas NF location dependencies, in conjunction with the limited geographic footprint of NF providers (NFPs), raise the need for NSE across multiple NFPs.**

In this paper, we present a holistic solution to the multi-provider NSE problem. We decompose NSE into (i) NF-graph partitioning performed by a centralized coordinator and (ii) NF-subgraph mapping onto datacenter networks. We present linear programming formulations to derive near-optimal solutions for both problems. We address the challenging aspect of traffic scaling by introducing a new service model that supports demand transformations. We also define topology abstractions for NF-graph partitioning. Furthermore, we discuss the steps required to embed service chains across multiple NFPs, using our NSE orchestrator (Nestor). We perform an evaluation study of multi-provider NSE with emphasis on NF-graph partitioning optimizations tailored to the client and NFPs. Our evaluation results further uncover significant savings in terms of service cost and resource consumption due to the demand transformations.

***Index Terms*—Network Function Virtualization, network service embedding, service chaining, orchestration.**

I. Introduction

Middleboxes have become an indispensable component of the network infrastructure. Middleboxes, such as firewalls, intrusion detection systems, redundancy elimination boxes, load balancers, proxies, application gateways, are prevelant in enterprise networks, satisfying the increasing needs of network operators in terms of security and access control [40]. Despite their widespread adoption, middleboxes exhibit significant limitations in terms of customization, resource efficiency, and manageability [39], [40]. These limitations mainly stem from the fact that middleboxes are built of specialized hardware; in other words, a middlebox cannot be repurposed for another packet processing functionality. This essentially leads to appliance sprawl, and, in turn, to substantial capital (CAPEX) and operational expenses (OPEX) for enterprises.

To mitigate some of these problems, Network Function Virtualization (NFV) promotes the deployment and consolidation of network functions (NF) on platforms built of commodity components [3], [5], [7], [6], [8]. This can lead to a reduction in OPEX and CAPEX, either by deploying consolidated software middleboxes in virtualized network infrastructures owned by the enterprise (*i.e.,* private clouds) or by outsourcing NFs to public clouds, based on emerging cloud computing models, such as NF-as-a-service (NFaaS). The latter, especially, can result in substantially higher OPEX and CAPEX savings, since NFaaS obviates the need to acquire, deploy, and operate additional network appliances on clients' premises, whereas fault management and maintenance is also left to the cloud operator. In fact, NFaaS is under way, as (mainly tier-1) Internet Service Providers (ISPs) have started to use existing micro-datacenters (DCs) for NFaaS offerings to their clients [25]. NFaaS is expected to be even more appealing to enterprises as the deployment of micro-DCs expands, offering a wider range of NFV Points-of-Presence (PoP) to clients.

In this respect, NF-graphs expressing service chains (*i.e.,* ordered sequences of middleboxes) should be mapped onto datacenter networks, ensuring correctness and resource efficiency, while complying with NF Provider (NFP) policies (*e.g.,* minimizing the embedding footprint to generate more revenue in the long run). In fact, NFP policy may contradict with the client's objective function, *i.e.,* the client will primarily seek to minimize his expenditure whereas the provider's policy may not yield the "cheapest" embedding.

The network service embedding (NSE) problem is further exacerbated by the location dependencies of certain NFs (*e.g.,* proxies and caches should be placed in proximity to the enterprise network, while packet filters should be deployed close to traffic sources for increased bandwidth conservation at the event of DoS attacks) and the limited footprint of NFPs.

More precisely, a single NFP may not satisfy the location constraints of all NFs in a service chain (*i.e.,* the NFP may not have NFV PoPs close to all end-points of the service chain), raising the need for NSE across multiple providers. Such embeddings should satisfy the objectives of clients (*e.g.,* expenditure minimization) and providers (*e.g.,* revenue maximization), whereas embedding methods should address the intricacies of multi-provider aspects (*i.e.,* restrictions in the resource and network topology information disclosed by NFPs to third parties) [21]. Existing solutions for multi-provider virtual network (VN) mapping [21], [29] generate embeddings based on VN graphs with specific bandwidth demands on each edge. This highly abstract service model cannot represent service chain resource demands, given the fact that certain NFs (*e.g.,* caches, redundancy elimination boxes) may compress or amplify traffic, raising the need for resource reservations substantially different from the demands specified in the chain.

In this paper, we present a holistic approach to multi-provider NSE, addressing these issues. In particular, we propose a network service embedding orchestrator (Nestor), which generates efficient embeddings via network graph rendering, service chain partitioning among DCs, and chain segment mappings onto the DC networks. Nestor decouples service chain partitioning from NFPs by interposing a network service composition layer (NSCL) between the clients and the NFPs. Essentially, NSCL comprises a separate layer that handles the partitioning of service chains among NFPs, eliminating the need for negotiation and contracting between individual clients and multiple NFPs. Instead, a client merely needs to establish a contract with a single NSCL for multi-provider NSE. Such a layer is also employed in network virtualization architectures, *e.g.,* in GENI [4], 4WARD [1], [38], and CABERNET [47].

In this respect, our contributions are as follows:

- We derive linear programming formulations for service chain partitioning. We particularly provide two variants of a linear program (tailored to the client or the NFP) to assign NFs to DCs and subsequently generate NF-subgraphs (*i.e.,* request segments) mappable to DC networks, by employing virtual gateways for inter-segment traffic aggregation. To facilitate service chain partitioning, we define a topology abstraction that conceals all information that is deemed confidential by NFPs.
- We introduce a new service model to simplify the specification of network service requests and deal with traffic-scaling NFs. With respect to the latter, the service model supports CPU and bandwidth demand transformations in the service chain.
- We design a linear program for the mapping of service chain segments onto DC networks (carried out by NFPs).
- We perform an evaluation study that sheds light into the tussles between contrasting provider and client objective functions.

In particular, the service model and service chain partitioning comprise novel aspects, as existing work on NSE does not take into account the traffic scaling caused by certain NFs, whereas most NSE methods are restricted to service mapping onto one or multiple NFV PoPs owned by the same NFP [19], [35], [37]. This paper extends our previous work in [20], by providing linear programming formulations for service chain partitioning and segment mapping to reduce the time complexity of the integer linear programs presented in [20]. We also provide a more elaborate NSE problem description, additional evaluation results, and extensive related work discussion.

The remainder of the paper is organized as follows. Section II describes the NSE problem. In Section III, we introduce our service model and topology abstractions for NSE. Section IV provides an overview of Nestor and discusses the steps required for the embedding of a service chain across multiple NFPs. In Sections V and VI, we present methods for NF-graph partitioning and the mapping of NF-subgraphs onto DC networks, respectively. In Section VII, we present our evaluation results and discuss the efficiency of Nestor. Section VIII discusses related work. Finally, in Section IX, we highlight our conclusions.

II. Problem Description

Service chaining is a common abstraction for the expression of network service requirements [26], [36]. A service chain represents the exact sequence of NFs traversed by one or multiple flows. Fig. 1 illustrates an example of service chaining. In this example, two different groups of enterprise network users at one site (*e.g.,* front-desk and sales) access a web server cluster and a database server residing in another site. Traffic from both groups traverses a cache, firewall, and a redundancy elimination (RE) appliance, whereas the traffic of "Group A" is sent through a load balancer and a web application firewall.

A service chain can be expressed as an NF-graph. Each vertex in the graph represents an NF or an end-point, whereas each edge represents a virtual link connecting a pair of NFs or an NF with an end-point. Each vertex representing an NF is associated with a CPU demand and, possibly, with a location constraint (*e.g.,* relative to the location of an end-point). Each edge is also associated with a bandwidth demand.

NSE consists in mapping such NF-graphs across multiple DCs, such that all location constraints and resource demands are satisfied (Fig. 2). NSE optimization is subject to provider policies and Service Level Agreements (SLA) between NFPs and clients. In this paper, we are dealing with service mapping across DCs operated by multiple NFPs, since the footprint of individual NFPs may not satisfy the location dependencies of all NFs in the NF-graph.

In this respect, we decompose multi-provider NSE into two sub-problems: (i) NF-graph partitioning among DCs and (ii) NF-subgraph mapping onto a DC network. This approach to the NSE problem is mainly driven by the multi-provider intricacies. More specifically, we expect that NFP policies will be governed by today's ISP policies, *i.e.,* NFPs will restrict information disclosure and interoperability with third parties and especially competitors, such as other NFPs. Hence, NF-graph partitioning should be performed based on an abstract network view, *i.e.,* topology information which is not considered confidential by NFPs (we discuss such topology

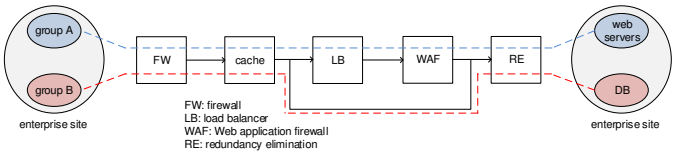


Fig. 1. Service chain example.

abstractions in Section III-B). To this end, we interpose a network service composition layer (NSCL) between the clients and the NFPs. In particular, the NSCL receives service chain specifications (*i.e.,* NF-graphs) from the client and partitions each NF-graph into NF-subgraphs assigned to separate DCs. We consider NSCL as a separate role from the NFP. As such, the NSCL is likely to be operated by a different organization. However, we do not preclude a single organization (*e.g.,* an ISP) fulfilling the roles of a NSCL and an NFP at the same time.

In the following, we delve into these NSE sub-problems from the perspective of policy and associated optimization objectives:

NF-graph partitioning. As NF-graph partitioning assigns NFs to DCs, NFPs will seek to balance the load across their DCs or will prefer to use DCs that can be reached through underutilized paths. In Section III-B, we discuss how NFPs can disclose DC preferences to the NSCL, allowing NSCL to optimize NF-graph partitioning according to the NFP policy (*e.g.,* DC load balancing). Furthermore, in Section VII-B we show that this NF-graph partitioning optimization generates more revenue for NFPs in the long run. On the other hand, a client will be mainly interested in minimizing his expenditure. Assigning NFs to underutilized DCs may not necessarily generate "cheap" embeddings for the client, as these DCs may have to be reached through longer and/or more expensive paths. The tussles between these different objective functions are of particular interest in our study. In this respect, in Section VII-B we investigate the impact of these two different objective functions on client's expenditure and NFPs' cumulative revenue.

NF-subgraph mapping. This consists in mapping NF-subgraphs (generated in the previous step) onto DCs networks (*i.e.,* the placement of NFs onto servers and the assignment of NF-subgraph edges onto physical network paths). The mapping should satisfy all resource requirements (*i.e.,* CPU, bandwidth) expressed in the NF-subgraph. Since the mapping is performed by the NFP (who has complete knowledge of the network topology and resource availability in his network), NF-subgraph mapping optimization will be based on the NFP policy. In this respect, we consider the minimization of the embedding footprint as the primary objective sought by the NFP. Since this minimizes the amount of resources reserved (as the heavily communicating components are placed in the same rack or server if possible), this will also incur low expenditure for the client. As such, in contrast to NF-graph partitioning, the objective functions of the NFP and the client are inline.

III. Service Model and Topology Abstractions

In this section, we discuss two critical aspects of multi-provider NSE: (i) the suitability of existing service models for service chain specification and embedding, and (ii) NFP policy-compliant topology abstractions for NF-graph partitioning.

A. Service Model

Existing VN embedding techniques (*e.g.,* [20], [29]) operate on a level of abstraction which is not suitable for NSE. The key difference lies in the particular information required for generating *correct* embeddings in both cases. In the case of VN embedding, existing solutions rely on VN graphs at which vertices represent virtual nodes, whereas edges express virtual links associated with a bandwidth demand. However, in the case of NSE, an embedding requires a NF-graph that takes the *order* and NF-specific *bandwidth demand transformations* into account.

These bandwidth demand transformations are associated with traffic-scaling NFs. More precisely, certain NFs, such as REs and caches, compress traffic, whereas other NFs (*e.g.,* packet multiplication, encryption) amplify traffic. The level of traffic scaling depends on various factors, such as the size and hit ratio for caches, the amount of duplicate content for RE, and the volume of traffic filtered by firewalls and intrusion detection systems (IDS). In this respect, Table I summarizes the traffic scaling of widely-used NFs, based on various studies [43], [12], [44]. A traffic scaling factor less than 1 implies traffic compression, whereas a traffic scaling factor greater than 1 entails traffic amplification.

Traffic-scaling NFs introduce complexity in NSE, as the bandwidth reservation on the ingress or egress link of a traffic-scaling NF may have to be different from the bandwidth demand specified in the chain. This effect will further propagate to subsequent NFs in the chain. For instance, assume traffic from a remote web server is sent through a firewall and a web cache (both hosted and operated by a NFP on behalf of a client), before it reaches a client. Web caching will reduce the bandwidth required both at the ingress link of the cache and the firewall.

Traffic-scaling NFs raise the need for demand transformations not only for bandwidth but also for CPU. More

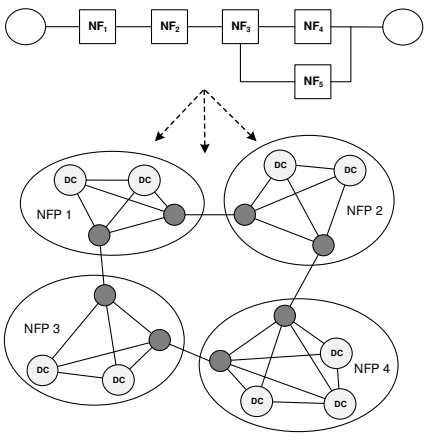


Fig. 2. Network service embedding.

TABLE I
TRAFFIC SCALING BY NETWORK FUNCTIONS.

Network function	Traffic scaling	Traffic scaling factor (φ)
Flow monitoring	No	-
Load balancer	No	-
NAT	No	-
RE	Yes	40–70% [43], 59–74% [12]
VPN (IPsec)	Yes	105–228% (for 64–1500-byte packets) [44]

specifically, the CPU demand for each NF can be derived based on the inbound traffic rate and the resource profile of each NF (*i.e.,* CPU cycles per packet). Resource profiles are available for a wide range of NFs [23], [22], while existing profiling echniques [45] can be applied to any flow processing workloads whose computational requirements are not known. Traffic scaling will affect the CPU resource reservations in the subsequent NFs in the chain. In the previous example, web caching will lower the CPU cycles needed for the firewall, since a fraction of HTTP traffic will be fetched from the cache.

In the following, we define a service model that enables resource demand transformations, simplifying the estimation of CPU and bandwidth demands. To this end, we initially introduce the traffic scaling factor φ_p^i, which is defined as the ratio of outbound traffic at port p of NF i over the aggregate inbound traffic at all ports. We particularly use the scaling factor per output port, since traffic may be split between multiple output ports depending on the outcome of packet processing. Our network service model consists of a NF-graph at which each vertex (corresponding to an NF) i is associated with a traffic scaling factor φ_p^i per port p (Fig. 3). Essentially, φ_p^i is used for the estimation of the bandwidth demand over each link, given the aggregate inbound traffic rate at each NF. The adjustment of φ_p^i for a traffic-scaling NF can be derived based on traffic statistics from middleboxes with the same functionality, deployed on the client's premises. In case such information is not available, φ_p^i can be adjusted based on statistics available from middlebox studies [43], [12], [44] or other network operators. Since achieving a very accurate estimation of φ_p^i may be difficult, φ_p^i can be set to the lowest traffic compression level or the highest level of traffic amplification (assuming a known range of traffic scaling, as shown in Table I). This approach ensures that bandwidth allocation will be sufficient, while any spare bandwidth can be distributed proportionally to the clients. After φ_p^i has been adjusted for each NF in the service chain, the client simply needs to specify the rate of the traffic generated at each end-point. CPU demands do not need to be specified in this service model, as they can be directly computed based on the inbound traffic rate and the resource profile of each NF, as explained above.

B. Topology Abstractions

Topology abstractions are a prerequisite for NF-graph partitioning, since the NSCL will not have access to detailed substrate topology information. In this respect, we seek to identify topology abstractions that conceal any information deemed as confidential by NFPs. To this end, we take into account information disclosed by ISPs and cloud providers.

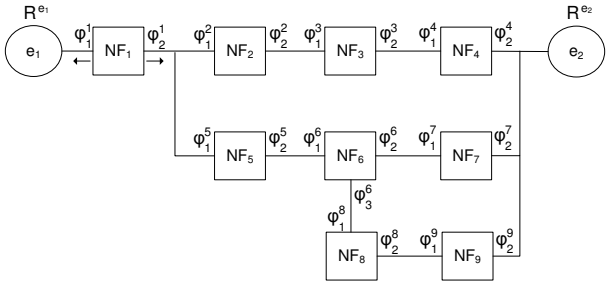


Fig. 3. Service model (R represents the traffic rate generated at an end-point and φ_p^i denotes the outbound to inbound traffic ratio at the port p of the NF i).

For example, ISPs often publish simplified PoP-level topologies [42], whereas cloud providers advertise resource types across different availability zones [2].

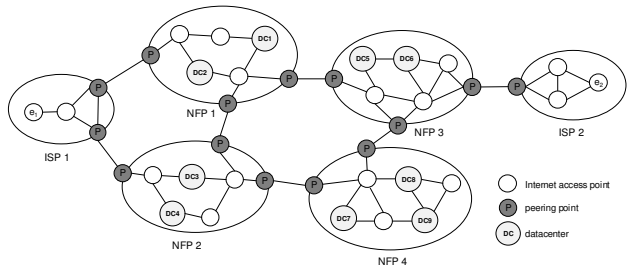
We depart from a PoP-level topology view that includes the Internet access points, NFV PoPs (*i.e.,* DCs), and peerings with neighbouring networks. Fig. 4(a) depicts such a topology spanning four NFPs and two ISPs. Since the end-points (*i.e.,* e_1, e_2) are fixed, we need a network view that simplifies the estimation of the link costs between the end-points and the DCs. Based on Fig. 4(a), we derive an abstract network view that obscures the Internet access points and represents (i) a full mesh of DCs and peering nodes within each NFP (each link connecting a DC with a peering node or a pair of peering nodes corresponds to the shortest path between the respective nodes), and (ii) the peerings among NFPs (Fig. 4(b)). This topology abstraction combined with NF and link costs provides the required information for the estimation of the overall embedding cost.

The edges of this topology graph can be annotated with weights assigned by each NFP, according to the NFP's policies (*e.g.,* load balancing), similarly to the Multi Exit Discriminator (MED) attribute of the Border Gateway Protocol (BGP). A NFP may wish to incorporate DC utilizations into the weights of the adjacent links, avoiding the explicit advertising of DC utilization information. We particularly consider a link weight offset which is dynamically adjusted according to the DC utilization level. Link weights are used by our NF-graph partitioning formulation variant which is tailored to NFPs (Section V).

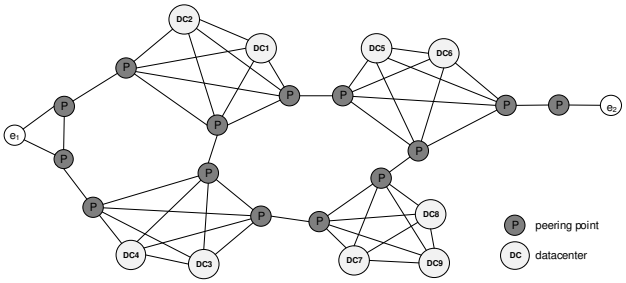
IV. NESTOR OVERVIEW AND MODEL

In this section, we provide an overview of Nestor and discuss the sequence of steps for service chain embedding. We further introduce the network model used in our formulations in the following sections. Nestor processes and embeds NF-graphs specified based on the service model presented in Section III-A. The topology abstraction in Section III-B represents the view of the NSCL on the substrate network topologies. To embed service chains, Nestor implements a NSE control plane, which is distributed across the NSCL, the NFPs, and the DCs deployed by each NFP. In our embedding framework, we assume trustworthy NFPs that disclose correct resource and topology information.

In the following, we discuss the steps for service chain embedding by Nestor:

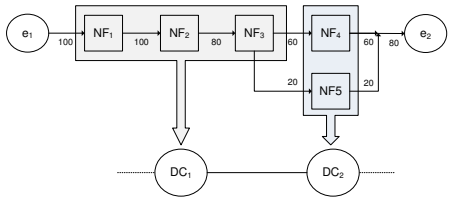


(a) PoP-level topology

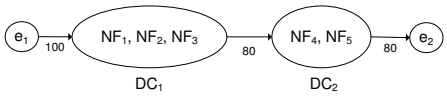


(b) Topology abstraction

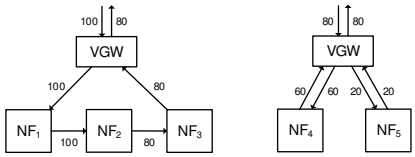
Fig. 4. Topology abstraction for request partitioning.



(a) Assignment of NFs to DCs



(b) Bandwidth demand aggregation



(c) Generation of subgraphs with virtual gateways (VGW)

Fig. 5. NF-graph partitioning.

Graph Rendering. Graph rendering consists in the transformation of detailed topology graphs into topology abstractions that facilitate NF-graph partitioning while obscuring any confidential information for NFPs. Each NFP generates the topology abstraction for his own network and subsequently annotates the edges of the graph with the link costs (*e.g.,* cost per bandwidth unit) and optionally with weights representing link and DC preferences. The NSCL collects the graphs from all participating NFPs and stitches them together constructing an abstract network view that spans all NFPs (*i.e.,* Fig. 4(b)). Since NFPs may adjust link weights differently or pick weight values from different intervals, the NSCL uses *min-max normalization* to normalize the weights advertised by each NFP. This ensures that weights across different NFPs are comparable, when the NSCL partitions NF-graphs using weight minimization (see below). New topology abstractions are generated upon significant substrate topology changes or the participation of new NFPs[1]. Link weights are updated on the existing network graphs in response to changes in resource utilization levels or NFP policies.

NF-Graph Partitioning. NF-graphs are partitioned among NFPs, when there is no single NFP that satisfies the location dependencies of all NFs in the service chain. More precisely, the NSCL identifies a list of candidate DCs[2] for each requested NF by matching NF location constraints against each NFP's footprint. Subsequently, the NSCL uses two variants of a linear program (LP) for NF-graph partitioning, tailored to (i) the client (*i.e.,* expenditure minimization) or (ii) the NFPs (*i.e.,* weight minimization) based on the weights disclosed by NFPs.

[1]In our implementation, the amount of data required to encode the data structure used for the NFP topology (with 5 DCs), peerings, costs, and weights into an XML message is 2.6-2.8 KB.

[2]DCs with cheap NF offerings and scarce resources may lead to request rejections, as such DCs are likely to be chosen for NF placement during the NF-graph partitioning step. This problem can be rectified by requiring from NFPs to stop advertising resources from highly utilized DCs.

The NF-graph partitioning formulations are discussed in detail in Section V.

NF-graph segments are computed by the LP solver output with NF-to-DC assignments (Fig. 5(a)). First, the NSCL computes the total inbound and outbound bandwidth demand for each segment (Fig. 5(b)). Next, the NSCL generates a NF-subgraph, at which all inter-segment traffic traverses a virtual gateway (VGW), as shown in Fig. 5(c). This NF-subgraph allows the binding of the VGW with the DC network gateway, augmenting the mapping of each subgraph onto the assigned DC.

NF-Subgraph Mapping. Each NF-subgraph is mapped onto the assigned DC network by the corresponding NFP. This process does not require any topology abstractions, since each NFP has a complete view of the DC network topologies and the utilization of servers and links. We particularly consider 2-level hierarchical DC network topologies that provide sufficient capacity for data transfers between the few hundreds of servers deployed within each micro-DC. Nevertheless, our NF-subgraph mapping methods are also applicable to 3-layer fat-tree topologies, used for larger DCs. We assign NF-subgraphs to DC networks using an LP, which is discussed in detail in Section VI. The objective of NF-subgraph mapping is the minimization of the embedding footprint. We note that Nestor can accommodate NF-subgraph mapping methods with different objectives, in case an NFP wishes to exercise a different policy (*e.g.,* load balancing). Any such deviation from the mapping method promoted by Nestor is not expected to have system-wide implications. Instead, it will mainly affect the resource efficiency of the NFP that uses his own mapping algorithm.

In the following, we introduce models for the service chain requests and the substrate network. Both models are based on the abstractions presented in Section III.

Request Model. We use a directed graph $G_F = (V_F, E_F)$ to express a service chain request. The set of vertices V_F includes

all NFs and the end-points that comprise the request. Each NF i is associated with an traffic scaling factor per port p, denoted by φ_p^i. Each end-point is associated with a traffic generation rate, which, combined with φ_p^i, gives the bandwidth demand d^{ij} for each edge $(i,j) \in E_F$. The computing demand d^i of each NF is estimated based on the inbound traffic rate and the NF resource profile (*i.e.,* CPU cycles / packet).

Substrate Network Model. We specify topology abstractions (Section III-B) and substrate network topologies using an undirected graph $G_S = (V_S, E_S)$. We use α_u and β_{uv} to express the monetary cost of NFs and links, respectively. As discussed in Section III-B, each graph edge $(u,v) \in E_S$ is associated with a weight, denoted by w_{uv}, which is assigned by the NFP. Furthermore, substrate nodes and links are associated with their residual capacity, represented by r_u and r_{uv}, respectively. We use λ^i to denote the distance tolerance of NF i, derived from the NF location dependence. To enforce NF location constraints, *i.e.,* to take the limited geographic footprint of NF providers into account, we further introduce l_u^i which represents the distance between the preferred location (*e.g.,* close to an end-point) and the DC u assigned to NF i. With a slight modification to the definitions of λ^i, l_u^i, distance tolerance could be also expressed in terms of number of hops, if such information is disclosed. A list of all notations is given in Table II.

TABLE II
NOTATIONS.

Symbol	Description
α_u	monetary server cost at DC u in $\$/GHz$
β_{uv}	monetary cost of link (u,v) in $\$/Mbps$
d^i	computing capacity demand of NF i in GHz
d^{ij}	bandwidth demand of edge (i,j) in $Mbps$
E_F	set of service chain links
E_S	set of links of a substrate topology
f_{uv}^{ij}	flow demand of edge (i,j) assigned to the intra-DC link (u,v) in $Mbps$
φ_p^i	outbound/inbound traffic ratio per port p for NF i
l_u^i	distance between the preferred location and the DC u assigned to NF i in km
λ^i	distance tolerance of NF i in km
r_u	residual capacity of server u in GHz
r_{uv}	residual capacity of link (u,v) in $Mbps$
V_F	set of NFs and end-points of a service chain request
V_S	set of DCs, peering nodes, end-points of a substrate topology or servers and switches of a DC
w_{uv}	weight of link (u,v)
x_u^i	assignment of NF i to DC or server u
y_{uv}^{ij}	mapping of NF graph edge (i,j) onto PoP-level graph edge (u,v)
z_u	assignment of any NF to server u

V. NF-GRAPH PARTITIONING

In this section, we present our NF-graph partitioning formulations. We initially derive an integer linear programming (ILP) formulation (Section V-A), which is subsequently transformed into a linear programming (LP) formulation, using relaxation and rounding techniques (Section V-B). In Section V-C, we empirically quantify the suboptimality of the LP formulation relative to the ILP formulation and discuss the trade-off between optimality and solver runtime for NF-graph partitioning.

A. Integer Linear Programming Formulation

Following the discussion on NF-graph partitioning objectives in Section II, we consider two ILP variants that reflect the point of view of the client and of the NFPs, respectively. The key difference in the formulations lies in the objective function. For the client side, we minimize the overall service cost (*i.e.,* the client expenditure), whereas for the NFPs we seek to balance the load across DCs within the NFP's network. More specifically, the first objective function, denoted as **Min-C**, minimizes the overall monetary cost for the client, by accumulating all the monetary NF and link costs. On the other hand, the objective function **Min-W** minimizes a weighted version of the overall network utilization, which essentially incentivizes network-wide load balancing. The corresponding link weights express a convex function of the utilization of links and DCs mimicking, for example, the impact of utilization on the average delay in simple queueing models such as the M/G/1 model [17]. Fig. 6 depicts examples of weight adjustments based on DC utilization. In principle, we consider that these weights will be adjusted by NFPs based on their policy, similar to the link weight adjustment performed by ISPs for intra-domain routing.

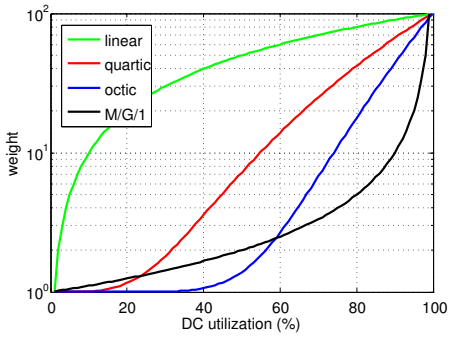


Fig. 6. Examples of different convex link weight adjustment functions that range from a simple linear mapping to an M/G/1 queueing inspired mapping.

In the ILP formulations, we use the binary variable x_u^i to express the assignment of NF i to the DC u. Similarly, the binary variable y_{uv}^{ij} indicates whether the NF-graph edge $(i,j) \in E_F$ has been mapped onto the PoP-level graph edge $(u,v) \in E_S$. The two NF-graph partitioning ILP formulations are given in the following, where both resort to the same constraints (3)-(7):

Min-C:

$$\text{Minimize} \sum_{u \in V_S} \alpha_u \sum_{i \in V_F} d^i x_u^i + \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} \beta_{uv} \sum_{(i,j) \in E_F} d^{ij} y_{uv}^{ij} \tag{1}$$

OR

Min-W:

$$\text{Minimize} \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} w_{uv} \sum_{(i,j) \in E_F} d^{ij} y_{uv}^{ij} \tag{2}$$

subject to:

$$\sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_F \tag{3}$$

$$\sum_{\substack{v \in V_S \\ (u \neq v)}} (y_{uv}^{ij} - y_{vu}^{ij}) = x_u^i - x_u^j$$

$$i \neq j, \forall (i,j) \in E_F, \forall u \in V_S \tag{4}$$

$$l_u^i x_u^i \leq \lambda^i \quad \forall i \in V_F, \forall u \in V_S \tag{5}$$

$$x_u^i \in \{0,1\} \quad \forall i \in V_F, \forall u \in V_S \tag{6}$$

$$y_{uv}^{ij} \in \{0,1\} \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{7}$$

Next, we briefly explain the ILP constraints. Constraint (3) ensures that each NF $i \in V_F$ is mapped exactly to one DC. Condition (4) preserves the binding between the NF and the link assignments. More precisely, this condition ensures that for a given pair of assigned nodes i, j (*i.e.,* NFs or end-points), there is a path in the network graph where the edge (i, j) has been mapped. Condition (5) enforces NF location constraints. Finally, the conditions (6) and (7) express the binary domain constraints for the variables x_u^i and y_{uv}^{ij}, *i.e.,* the assignment of NFs to DCs and the mapping of NF edges to PoP-level graph edges. In addition, we require fixing the end-points k in the request to the respective locations u by setting $x_u^k \leftarrow 1$.

B. Linear Programming Formulation

In the following, we reduce the time complexity of the preceding NF-graph partitioning ILP formulation. To this end, we transform the ILP into an LP formulation by relaxing the integer domain constraints (6) and (7), as follows:

$$x_u^i \geq 0 \quad \forall i \in V_F, \forall u \in V_S \tag{8}$$

$$y_{uv}^{ij} \geq 0 \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{9}$$

Now, the NF-graph partitioning LP formulation consists of the objective functions (1) or (2), the constraints (3) – (5) from the original ILP formulation and the two new domain constraints (8) and (9). The relaxation of (6) and (7) requires additional steps to preserve the correctness of the other problem constraints.

To this end, we introduce a rounding algorithm (Algorithm 1) to derive near-optimal solutions for the NF-graph partitioning problem. Algorithm 1 iteratively steps through all the remaining mapping combinations that contain non-integer values. The algorithm continuously chooses mapping combinations which have the highest x_u^i values, which can be interpreted as a high probability of successful mapping, *i.e.,* choosing max x_u^i. This mapping combination is fixed for the following iterations of the algorithm if the location constraint is satisfied. The algorithm stops when a solution with only integer values is found and returns a near-optimal NF-graph partitioning solution or it terminates with a rejection of the request if no more solutions can be found.

C. NF-Graph Partitioning Comparison

Next, we compare the ILP and LP-based partitioning in terms of the two objective functions, *i.e.,* cost minimization for *Min-C* and weight minimization for *Min-W*. To this end, we partition 25K service chains among 50 DCs. The rest of the evaluation parameters are shown in Table III (this microbenchmark is independent of the evaluation in Section VII).

Fig. 7 illustrates the normalized resource unit costs (*i.e.,* for CPU, bandwidth) after NF-graph partitioning with *Min-C*. According to Fig. 7, CPU cost is almost equal for both variants (ILP and LP), whereas LP yields 4.4% higher median bandwidth cost compared to the ILP solution. Similarly, Fig. 8 illustrates the sum of weights associated with ILP and LP, in the case of *Min-W*. NF-graph partitionings with ILP and LP achieve a mean sum of weights per service chain of 358.5 and 360.5, respectively. As such, the LP variant yields only marginal suboptimality compared to the original ILP. However, the LP solver runtime is one magnitude lower than the ILP solver runtime, irrespective of the substrate network size (Fig. 9).[3] This outweighs the possible LP suboptimality.

Algorithm 1 Request Partitioning with LP

```
repeat
    {x_u^i, y_uv^ij} ← Solve_LP(..)
    X ← {x_u^i | x_u^i ∉ {0,1}}
    if X ≠ ∅ then
        {i_fx, u_fx} ← argmax_{i∈V_F, u∈V_S} X
        if l_{u_fx}^{i_fx} ≤ λ^{i_fx} then
            Add_LP_Constraint("x_{u_fx}^{i_fx} = 1")
        else
            Add_LP_Constraint("x_{u_fx}^{i_fx} = 0")
        end if
    end if
until (X = ∅) ∨ NoFeasibleSolutionLP
return {x_u^i, y_uv^ij}
```

VI. NF-Subgraph Mapping

In this section, we discuss the NF-subgraph mapping. We first present a mixed integer programming (MIP) formulation in Section VI-A. Subsequently, we transform this MIP model into an LP model to reduce the MIP solving time complexity (Section VI-B). In Section VI-C, we compare the two variants (MIP and LP) in terms of optimality and solver runtime.

A. Mixed Integer Programming Formulation

In the following, we derive a MIP formulation for the problem of NF-subgraph mapping onto DC networks with the aim of minimizing the embedding footprint. This is essentially associated with the minimization of allocated servers and inter-rack traffic. Especially, the latter objective is critical for datacenters with high oversubscription ratios. Hence, we use the binary variable z_u to indicate whether *any* NF has been assigned to server u, *i.e.,* $z_u = 0$ when there is no NF assigned to server u; $z_u = 1$ otherwise. Note that the variable z_u depends on x_u^i, *i.e.,* the assignment of NF i to server u.

In the following, we use the variable d^{ij} to express the bandwidth demand between a pair i, j of NFs. In this context, the flow variable f_{uv}^{ij} denotes the amount of flow bandwidth (*i.e.,* in bandwidth units) the DC link (u,v) carries for the NF-graph edge $(i, j) \in E_F$. The MIP formulation has the objective function given in (10), which consists of two terms, *i.e.,* the number of assigned servers and the accumulated flow bandwidth divided by the total NF bandwidth demand.

[3] We conducted microtests with substrate topologies with 5, 10, and 15 NFPs, each one spanning 5 DCs. Tests were carried out on a server with Intel Xeon CPU at 2.53 GHz,.using a single CPU core.

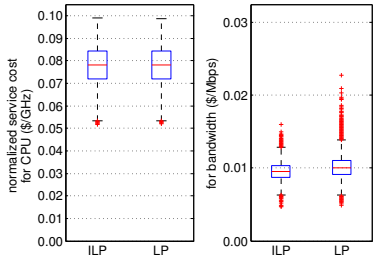


Fig. 7. Min-C: normalized resource cost.

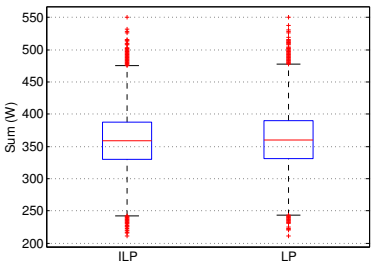


Fig. 8. Min-W: weight sum.

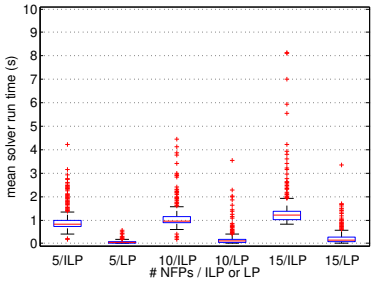


Fig. 9. ILP/LP solver run time for NF-graph partitioning (with a diverse number of NFPs).

Essentially, the second term yields 1 if all NF-graph edges $(i,j) \in E_F$ are mapped onto single-hop paths.

Minimize

$$\sum_{u \in V_S} z_u + \frac{1}{\sum_{(i,j) \in E_F} d^{ij}} \cdot \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} \sum_{(i,j) \in E_F} f_{uv}^{ij} \tag{10}$$

subject to:

$$\sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_F \tag{11}$$

$$\sum_{v \in V_S} f_{uv}^{ij} - \sum_{v \in V_S} f_{vu}^{ij} = d^{ij}(x_u^i - x_u^j) \quad i \neq j, \forall i,j \in V_F, u \neq v, \forall u \in V_S \tag{12}$$

$$\sum_{i \in V_F} d^i x_u^i \leq r_u \cdot z_u \quad \forall u \in V_S \tag{13}$$

$$\sum_{i,j \in V_F} f_{uv}^{ij} \leq r_{uv} \quad \forall (u,v) \in E_S \tag{14}$$

$$x_u^i, z_u \in \{0,1\} \quad \forall i \in V_F, \forall u \in V_S \tag{15}$$

$$f_{uv}^{ij} \geq 0 \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{16}$$

Next, we explain the constraints (11)–(16) of our MIP formulation. Condition (11) ensures that each NF $i \in V_F$ is mapped exactly to one server. Constraint (12) enforces flow conservation, *i.e.,* the sum of all inbound and outbound traffic in switches and servers that do not host NFs is zero. The constraints (13) and (14) ensure that the allocated computing and bandwidth resources do not exceed the residual capacities of servers and links, respectively. Condition (13) is further used for the binding between the two binary variables z_u and x_u^i. Finally, condition (15) expresses the binary domain constraint for the variables x_u^i and z_u, while constraint (16) ensures that the flows f_{uv}^{ij} are always positive. We further assume that the first element in V_F represents the virtual gateway which we bind to the physical gateway *GW* by setting $x_{GW}^{V_F(1)} \leftarrow 1$.

B. Linear Programming Formulation

In the following, we describe a transformation of the above MIP model to an LP model by relaxing the integer domain constraints. Specifically, we replace Equation (15) by:

$$x_u^i \geq 0 \quad \forall i \in V_F, \forall u \in V_S \tag{17}$$

$$z_u \geq 0 \quad \forall u \in V_S \tag{18}$$

Existing constraints could be violated after the relaxation of x_u^i and z_u thus yielding infeasible solutions. Such solutions can be iteratively excluded by re-running the LP solver after rounding variables. This iterative reduction of the solution space will finally exclude all non-binary solutions that are returned by the LP solver, similarly to the LP-based NF-graph partitioning of Section V-B. Nevertheless, the ILP formulation of the NF-subgraph mapping requires additional modifications. For $x_u^i < 1$ or if $z_u > 1$ the node capacity constraint (13) could be violated. On the contrary, if $z_u < 1$, nodes with sufficient capacity could be ignored which yields merely suboptimal but feasible solutions. Hence, we modify the domain constraint of z_u in (18) to:

$$0 \leq z_u \leq 1 \qquad \forall u \in V_S \tag{19}$$

while adding the following constraint

$$z_u \geq x_u^i \quad \forall i \in V_F, \forall u \in V_S, \tag{20}$$

which yields the correct number of servers in the objective function after the rounding of all corresponding x_u^i.

In conclusion, the LP formulation of the NF-subgraph mapping model consists of the objective function (10) and the constraints (11)–(14), (17), (19), (20).

LP solutions may contain non-binary values for x_u^i and z_u, which we handle using a rounding algorithm (Alg. 2). This algorithm iteratively fixes the most probable assignment of a NF to a server (*i.e.,* x_u^i) and repeats LP solver runs as long as there are feasible LP solutions and non-binary x_u^i. If a NF-subgraph mapping becomes infeasible due to the already assigned demand, the corresponding mapping will be precluded by setting $x_u^i \leftarrow 0$.

C. NF-Subgraph Mapping Comparison

Next, we investigate the suboptimality of the LP-based mapping compared to the MIP formulation in terms of NF-graph mapping efficiency. To this end, we assign 25K NF-graphs onto a DC, using both mapping variants. Each NF-graph contains 3 to 20 NFs. The remaining evaluation parameters used for this test are identical to the NSE evaluation in

Algorithm 2 NF-Subgraph Mapping with LP

repeat
  $\{z_u, x_u^i, f_{uv}^{ij}\} \leftarrow$ **Solve_LP(..)**
  $X \leftarrow \{x_u^i \mid x_u^i \notin \{0,1\}\}$
  if $X \neq \emptyset$ **then**
    $\{i_{fx}, u_{fx}\} \leftarrow argmax_{\{i \in V_F, u \in V_S\}} X$
    if $\sum_{i \in \{V_F | x_{u_{fx}}^i = 1\}} d^i + d^{i_{fx}} \leq r_{u_{fx}}$ **then**
      Add_LP_Constraint("$x_{u_{fx}}^{i_{fx}} = 1$")
    else
      Add_LP_Constraint("$x_{u_{fx}}^{i_{fx}} = 0$")
    end if
  end if
until $(X = \emptyset) \vee NoFeasibleSolutionLP$
return $\{z_u, x_u^i, f_{uv}^{ij}\}$

Section VII (see Table III). We note that this microbenchmark is independent of the evaluation in Section VII.

For comparison between the two variants we stress the system at a level at which the original MIP variant starts dropping incoming requests due to resource shortages. Therefore, we set the request arrival rate to 10 per hour which in turn yields a mean acceptance rate of 99.87% and 99.86% for the MIP and LP variant. We further aim at comparing the resource efficiency of both NF-graph mapping variants. In this respect, Fig. 10 illustrates the number of allocated servers and racks for the LP variant relative to the MIP variant. The LP formulation results in a marginally higher number of servers and a negligibly higher number of racks. This is plausible since the NF-graph mapping objective function aims at minimizing link cost implicitly by co-locating NFs preferably in a single server, if possible, or in a single rack otherwise.

We further investigate whether the LP variant generates additional traffic within the DC, compared to the MIP. According to Fig. 11, LP results in marginally higher volume of inter-rack traffic and a more perceptible increase (*i.e.,* 8 to 11%) in the traffic within the racks. However, rack traffic is less expensive than inter-rack traffic and as mentioned above, this does not significantly impact the request acceptance rate and, therefore, the generated revenue. Eventually, the LP variant yields only marginal suboptimality compared to the MIP variant, whereas the LP exhibits at least one order of magnitude lower runtime compared to MIP as shown in Fig. 12.[4] As such, similar to NF-graph partitioning, we employ the LP-based solution for NF-graph mapping.

VII. Evaluation

In this section, we assess the efficiency of multi-provider NSE with Nestor. We particularly consider an online scenario at which we process and embed incoming requests one by one, as they arrive. We mainly focus on NF-graph partitioning and particularly on the impact of different partitioning objectives on service cost, load balancing, request acceptance, and generated revenue. To this end, we rely on the two NF-graph partitioning LP variants introduced in Section V-B. Upon partitioning, the mapping of NF-subgraphs to DCs is computed using the LP presented in Section VI-B. In the following, we present our evaluation environment (Section VII-A) and discuss our evaluation results (Section VII-B).

[4]Tests were carried out on a server with 2.53 GHz Intel Xeon CPU, using a single CPU core.

A. Evaluation Environment

We have implemented an evaluation environment for multi-provider NSE in C/C++. Our implementation includes the Nestor NSE framework (Section IV), a service chain generator, and a substrate network topology generator. We rely on CPLEX as optimizer for our LP/ILP models. Below, we provide further details on the substrate network and service chain specifications, as used in our evaluations.

Substrate Network. We generated a PoP-level substrate topology with 12 NFPs covering a region of the size of California. The substrate spans 50 homogeneous DCs, each one containing 200 servers in 10 racks. For each DC, we have generated a 2-level hierarchical network topology. Additional evaluation parameters (*e.g.,* server/link capacities, resource pricing) appear in Table III. Resource pricing has been adjusted based on information collected from major cloud providers (*e.g.,* Amazon EC2 [2]) and Internet peering databases.

Service chains. Network service requests are generated based on service chain templates. These templates are composed of NFs that correspond to real middlebox applications (*e.g.,* firewall, load balancing, RE). Each NF is associated with a traffic scaling factor (φ), adjusted according to the statistics summarized in Table I. The NF computational requirements and bandwidth demands are derived from our network service model (Section III-A), given the φ adjustments and the traffic rate at the end-points. The traffic rate is randomly sampled from a uniform distribution. The end-points are randomly selected out of 50 possible locations with a minimum distance of 250km to each other.

We present evaluation results with (i) a very large number of non-expiring requests to assess efficiency at a wide range of utilization levels and especially when our system is under stress and (ii) expiring requests that arrive according to the Poisson distribution with different arrival rates (*i.e.,* 15, 20, 25 requests per min.) in order to investigate the system convergence to a steady state. Expiring requests are associated with a duration between 1 hour and 1 week, sampled from a uniform distribution.

We use the following metrics for the evaluation of NSE efficiency:

- **Service cost** represents the client's expenditure for the network service.
- **DC load balancing level** is defined as the maximum over the average server CPU load across the DCs. Lower values represent better load balancing, whereas a value of 1 designates optimal load balancing.
- **Acceptance rate** is the number of successfully embedded requests over the total number of requests.
- **Revenue** accumulates the CPU and bandwidth units allocated for service chain embedding.

B. Evaluation Results

We perform a comparative study between the two NF-graph partitioning variants (Section V), *i.e.,* embedding cost

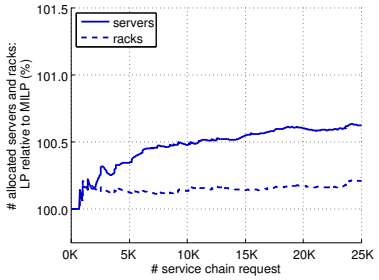


Fig. 10. Server and rack allocation.

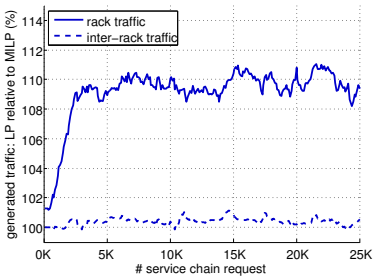


Fig. 11. DC traffic generation.

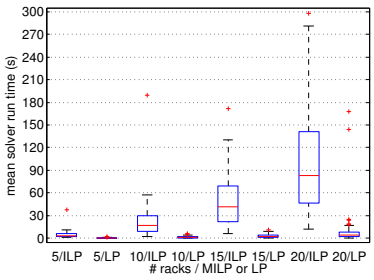


Fig. 12. MIP/LP solver run time for the mapping of NF-subgraphs to servers (with a diverse number of DC racks).

TABLE III
EVALUATION PARAMETERS.

Substrate network (PoP-level topology)	
NFPs / DCs	12 / 50
DCs per NFP	4–5
Peerings per NFP	2–4 (mean: 3.1)
Intra-domain link cost	unif. distrib. [0.002, 0.006] $/Mbps
Peering link cost	unif. distrib. [0.006, 0.018] $/Mbps
Server cost	unif. distrib. [0.05, 0.10] $/GHz
Inter-DC link capacity	100 Gbps
Substrate network (DC network topology)	
Root switches / racks per DC	5 / 10
Servers per rack	20
Server capacity	16 · 2 GHz
ToR-to-server link capacity	4 Gbps
Inter-rack link capacity	16 Gbps
Service chains:	
Number of NFs	uniform distrib. [10, 20]
Traffic generation rate	uniform distrib. [10, 100] Mbps

minimization (*Min-C*) and link weight minimization (*Min-W*). In addition, we use a *greedy* algorithm as baseline. This algorithm binds each NF with one of the end-points, depending on the NF location constraint (*e.g.,* NFs that are required to be close to the server or the client) or their order in the service chain (for NFs without location dependencies), and assigns each NF to the DC which is most proximate to the corresponding end-point. In case the closest DC does not have sufficient resources, the algorithm seeks to place the respective NF on the second most proximate DC and so forth, till all NFs have been assigned.

Fig. 13 illustrates the evolution of the cumulative service cost with 250K non-expiring requests. Both *Min-W* and the greedy algorithm yield a higher service cost than *Min-C*, which is optimized for service cost minimization. In particular, *Min-W* exhibits an increase in the service cost (relatively to *Min-C*) with the number of requests, eventually converging to 20% additional service cost, which is steadily incurred by the greedy algorithm.

The boxplots in Fig. 14 illustrate the decomposition of service cost into the CPU and bandwidth cost, normalized per resource unit. The lower service cost of *Min-C* stems from the significantly lower bandwidth cost (Fig. 14), considering that in absolute terms the fraction of bandwidth cost is one magnitude higher than the fraction of CPU cost. Essentially, *Min-C* achieves cost savings with the selection of DCs which are reachable over less costly paths. The greedy algorithm yields an average CPU cost of 0.075$/GHz, which corresponds to the average CPU cost across all NFPs, since DC selection is bound to randomly assigned end-points.

So far, *Min-C* appears very appealing for clients, since it minimizes their expenditure. However, *Min-C* may entail suboptimality for NFPs which we investigate in the following. In this respect, Fig. 15 depicts the evolution of load balancing level across the DCs. According to Fig. 15, *Min-W* converges to near-optimal load balancing after 100K requests, exploiting the DC utilization levels disclosed via the link weights. In comparison, *Min-C* yields worse load balancing. For instance, after 250K requests the highest server load is 5.3% and 18.2% above the average DC utilization, for *Min-W* and *Min-C*, respectively. On the other hand, the greedy algorithm yields a high degree of load imbalance, since it selects DCs close to the end-points.

Fig. 16 shows the request acceptance rates for the three NF-graph partitioning methods. Optimizing DC selection based on the disclosed weights (*i.e., Min-W*) inhibits the assignment of NF-subgraphs to highly utilized DCs, which usually leads to request rejections. As such, *Min-W* yields a higher request acceptance rate. Specifically, after 100K requests (which corresponds to a server utilization level of 80% across DCs), *Min-W* can embed 23% more requests than *Min-C*. On the other hand, the greedy algorithm suffers from a large number of rejections, due to the restrictions in DC selection. In this respect, Fig. 17 shows the acceptance rate versus the server utilization for the three partitioning variants. Fig. 17 corroborates the high resource efficiency of *Min-W* that exhibits the highest acceptance rate compared to *Min-C* and the greedy variant, while utilizing up to 88% of the server capacity.

Figs. 16 and 18 show a strong correlation between the acceptance rate and generated revenue. The LP variants generate substantially higher revenue from CPU and bandwidth, compared to the greedy algorithm. For *Min-W*, the highest acceptance rate is translated to a higher revenue, *i.e.,* up to 14% more than *Min-C*. This essentially designates *Min-W* as the preferred NF-graph partitioning method for NFPs.

In addition, we measure the acceptance rate of *Min-W* with 250K *expiring* requests and diverse arrival rates. Figs. 19(a) shows that acceptance rates converge to a steady state, irrespective of the arrival rate. Fig. 19(b) further depicts the acceptance rates for the three different arrival rates at steady state. In addition, a load balancing level between 1.1 and 1.4

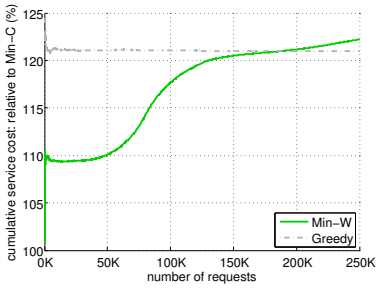


Fig. 13. Cumulative service cost with weight-minimized and greedy NF-graph partitioning relative to the cost-minimized partitioning.

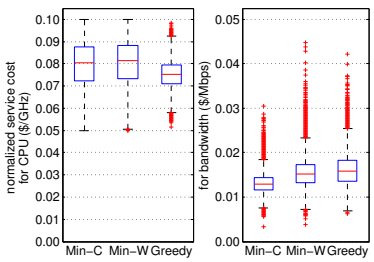


Fig. 14. Normalized service cost with cost-minimized, weight-minimized, and greedy NF-graph partitioning.

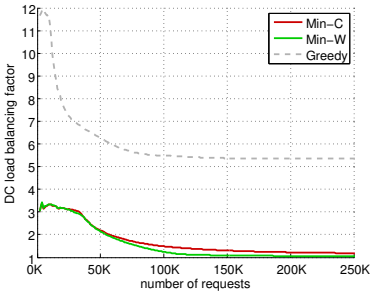


Fig. 15. Load balancing level with cost-minimized, weight-minimized, and greedy NF-graph partitioning.

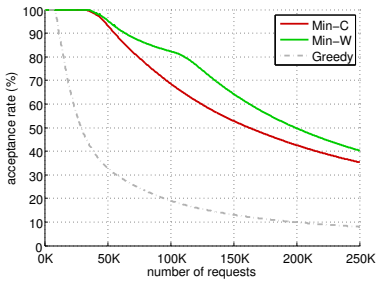


Fig. 16. Acceptance rate with cost-minimized, weight-minimized, and greedy NF-graph partitioning.

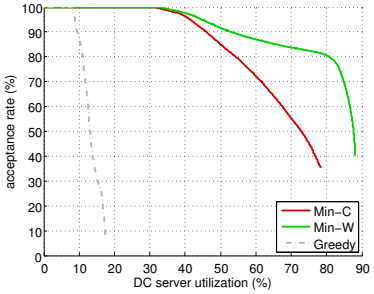


Fig. 17. Acceptance rate vs. server utilization with cost-minimized, weight-minimized, and greedy NF-graph partitioning.

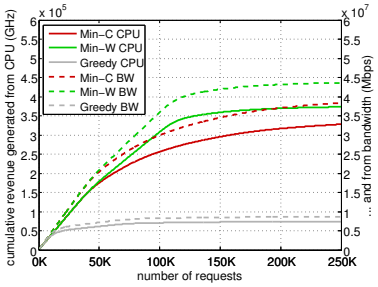


Fig. 18. Cumulative revenue with cost-minimized, weight-minimized, and greedy NF-graph partitioning.

is achieved, depending on the request arrival rate, as shown in Figs. 20(a) and 20(b). These results further indicate the efficiency of the proposed LPs for NF-graph partitioning and NF-subgraph mapping.

We also investigate the gains from the demand transformations with our service model (Section III-A). Specifically, we analyze the savings in terms of service cost and resource consumption using the *Min-C* and *Min-W* NF-graph partitioning variants, respectively. To this end, we process and embed the same set of requests (*i.e.,* service chains) with and without demand transformations. Each chain includes a random number of traffic-scaling NFs, and particularly NFs that compress traffic.

Fig. 22 depicts the average service cost per service chain with and without demand transformations. The two boxplots in Fig. 22 illustrate the fraction of service cost associated with CPU and bandwidth. We observe that demand transformations lead to 30% savings in the median bandwidth cost. This is translated into a significant reduction in the client's expenditure. There is also a perceptible CPU cost saving (Fig. 22), albeit lower than the respective bandwidth cost saving. The difference in the cost saving between CPU and bandwidth stems from the fact that traffic-scaling NFs affect the CPU demand of subsequent NFs in the chain, as opposed to traffic compression which occurs at the egress link of the traffic-scaling NF and propagates till the chain end-point.

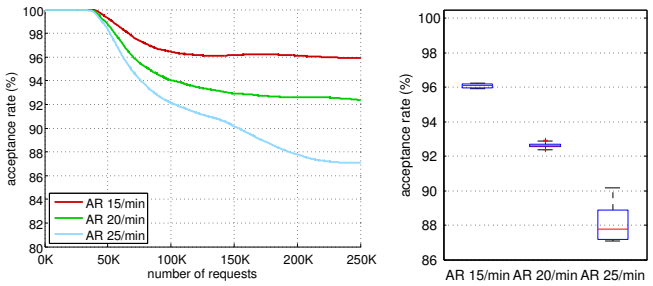
We further discuss the resource savings for the NFPs, due to the demand transformations in our service model. Fig. 23 consists of two boxplots that illustrate the CPU and overall bandwidth consumption with and without demand transformations. Similarly to Fig. 22, we observe a substantially lower bandwidth consumption and slightly lower CPU consumption when demand transformations are in use. These resource savings eventually lead to higher acceptance rates, as shown in Fig. 21. Essentially, coupling *Min-W* partitioning with demand transformations results in a significantly higher acceptance rate and, consequently, more revenue for the NFPs.

Finally, we investigate whether Nestor places NFs in the order specified in the service chain. In this respect, we measure the number of DC traversals by each embedding, *i.e.,* if each DC is traversed only once, this implies correctness. According to our results, in 89.3% of the embeddings, DCs are traversed only once, whereas in 10.5% of the embeddings, certain DCs are traversed twice. This occurs since chain partioning is optimized for embedding cost or link weight minimization (*i.e.,* enforcing correctness as a constraint could exclude cheap embeddings or lead to request rejections). Nevertheless, even in this case, correctness can be attained using various techniques such as tunneling, flow tagging [24], and NF-graph transformations [26], [16].

VIII. Related Work

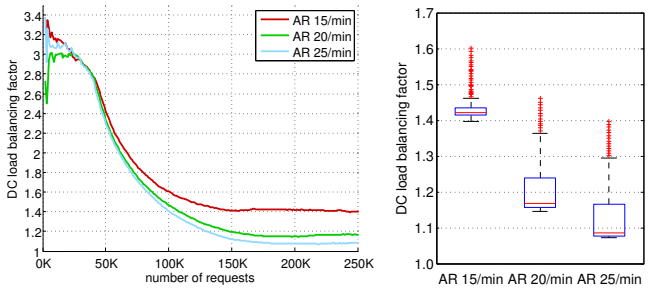
In this section, we discuss related work on NSE. We particularly discuss existing work on the two NSE aspects that we address in this paper, *i.e.,* (i) graph partitioning and (ii) NF-graph mapping onto a DC network.

Graph partitioning. Authors in [29] and our previous work [21] address the problem of multi-provider VN embedding. Both papers propose a LP for graph partitioning across multiple infrastructure providers (InPs). However, their



(a) Convergence (b) Steady state

Fig. 19. Acceptance rate with diverse arrival rates (AR) of expiring requests and weight-minimized NF-graph partitioning.



(a) Convergence (b) Steady state

Fig. 20. Load balancing level with diverse arrival rates (AR) of expiring requests and weight-minimized NF-graph partitioning.

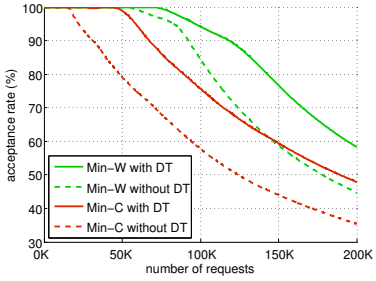


Fig. 21. Acceptance rate with and without demand transformations (DT).

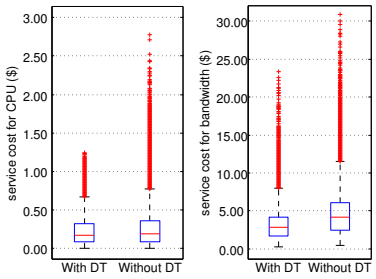


Fig. 22. Service cost per service chain with and without demand transformations (DT), both with Min-C.

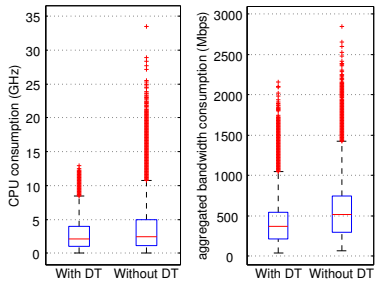


Fig. 23. Resource consumption per service chain with and without demand transformations (DT), both with Min-W.

approach is only valid for VN topology partitioning, as their request models cannot represent service chain specifications. Furthermore, VN-graph partioning is executed based on only an InP-level network view and the knowlegde of InP peerings, and as such, the VN-graph is partitioned among a set of InPs. In contrast, NF-graph partitioning generates NF-subgraphs mappable to DCs, and thereby, requires DC-level topology abstractions. Furthermore, Nestor provides NF-graph partitioning optimizations both the client and the NFP (*i.e.,* via the disclosure of link weights expressing NFP preferances for link and DC selection), as opposed to VN-graph partitioning in [29], [21], which is optimized only for the client (*i.e.,* expenditure minimization).

DistNSE [10] presents a distributed auctioning framework to partition NF-graphs across NFPs. DistNSE's main aim is to preserve service chain correctness (*i.e.,* a generated NF-subgraph should include a subset of NFs in the same order with the NF-graph) as well as the autonomy of each NFP (*i.e.,* each NFP should be allowed to implement his own policy in terms of NSE). In this respect, DistNSE does not seek to achieve NF-graph partitioning optimality. Instead, DistNSE enables NFPs to bid for NFs, according to their policy and resource availability.

NF-graph mapping. Most existing work on NSE focuses on mapping NF-graphs onto a single substrate network. The works in [35], [27], [11] present a solution to the NF-graph mapping problem, while considering different optimization goals. The authors further define a model for NF-graph transformations (*i.e.,* NF reordering, replication, or merging) to optimize the NF placement for the provider. Authors in [37] investigate gains in terms NF-graph mapping by NF decomposition (*i.e.,* breaking down NFs into a set of packet processing elements to facilitate mapping). Cohen at el. [19] formulate NF placement as a facility location and generalized assignment problem (GAP), without taking correctness into account. The authors further investigate several variations of the NF placement problem and propose approximation algorithms aiming at latency and NF setup cost minimization.

Authors in [32] tackle a variant of the NF mapping problem, *i.e.,* the placement of NFs on a network while ensuring that each path between a pair of end-points has at most one NF assigned. The proposed approximation algorithm further facilitates the incremental deployment of NFs, such that additional NFs can be rolled out without changes in NF placements. One aspect which has not been taken into consideration in this work is service chaining. In [33], the authors study the online variant of the service mapping problem. In this respect, they propose an exact method (based on an ILP) that maximizes the request acceptance rate while fulfilling constraints in terms of path length for the servic chain. Bari at el. [14] derive an ILP formulation and heuristic algorithm for service mapping with the objective of operational cost (*i.e.,* energy cost, traffic redirection cost and extra latency) and resource fragmentation minimization. To simplify NF-to-node mapping, the authors express each computing node capacity in terms of NF slots.

MIDAS [9] proposes an architecture for the coordination of on-path flow processing setup, assuming the wide-scale deployment of middleboxes in the network. In terms of NF placement, MIDAS uses a heuristic algorithm for order-preserving NF assignment, *i.e.,* a first-fit placement followed by an order-preserving worst-fit generates the NF assignment per service chain. STRATOS [26] and CloudNaaS [16] propose heuristic mapping algorithms that seek to minimize inter-rack traffic within DC networks. A similar approach is also

taken by Oktopus [13], SecondNet [28] and CloudMirror [30] for the assignment of virtual clusters to DCs. The authors in [48], [34], [41] present frameworks for NFV orchestration in different settings, *i.e.,* over different providers, for 5G mobile networks and over joint cloud and network resources, respectively. In addition, [48] presents a framework for the computation of all feasible mappings of service chains across multiple providers. The authors show a practically acceptable scaling behavior in the number of NFs despite an exponential growth in the local computation time.

Additional studies have tackled the problem of embedding VN- or NF-graphs onto a shared substrate network, relying on heuristic algorithms [46], [49], [31], [15] or linear programs [18], [21], [29]. However, these embedding methods are designed for arbitrary virtual and substrate network topologies, and, hence, are not optimized for NF-graph mapping onto DC networks. The work in [15] is a heuristic approach for a coordinated composition and embedding of non-expiring service chains. Furthermore, existing VN request models can not express bandwidth demand transformations which are required by traffic-scaling NFs.

Compared to all these approaches, Nestor provides a holistic solution for the NSE problem across multiple NFPs, including NF-graph partitioning, rendering of NF-subgraphs mappable to DCs, and NF-subgraph mapping. According to our knowledge, Nestor is the first NSE framework that takes traffic-scaling NFs into account, introducing demand transformations that alleviate the traffic scaling effects. Our evaluation study is focused on multi-provider aspects (*i.e.,* NF-graph partitioning) and more specifically, on the impact of constrasting provider and client objectives on NSE efficiency in terms of request acceptance rate, generated revenue, service cost, and network load balacing.

IX. Conclusions and Future Work

In this paper, we presented Nestor, a NSE orchestrator, that addresses the main challenges faced by the mapping of service chains across multiple NFPs. In this respect, we decomposed NSE into two problems (*i.e.,* NF-graph partitioning and NF-subgraph mapping), which we studied separately. For both problems, we presented ILP formulations, and subsequently, derived LP-based solutions for reduced time complexity and better scalability. Especially for NF-graph partitioning, which is the main focus of our study, we provided ILP/LP variants optimized for the client and the NFP. This allowed us to assess the impact of different partitioning optimizations on embedding efficiency. In this respect, we uncovered a trade-off between service cost minimization and revenue maximization. In particular, service cost minimization can potentially lead to cheaper NFaaS offerings, attracting more clients, but at the same time generates suboptimal embeddings that restrict the revenue of NFPs. Conversely, partitioning optimizations driven by NFP policies yield resource efficiency, maximizing the NFPs' revenue, but also entail more expensive and, thus, less competitive NFaaS offerings.

One of the most novel aspects of our work is the introduction of demand transformations, as a feature of our new service model for the specification of NF-graphs. The CPU and bandwidth demand transformations alleviate the traffic scaling effects, especially as they propagate across the NFs in the service chain. Our evaluations show significant resource and service cost savings, from which both the NFPs and the clients benefit. The gains are particularly high for the NFPs who can increase their revenue by embedding 10–15% more service chains, according to our evaluation results.

In future work, we will investigate efficient techniques for scaling existing service chain embeddings due to evolving demands and changes in service chain specifications. This essentially poses the need for coordination of NF placement, NF state transfer, and netwok updates, to minimize configuration overhead and network service disruptions.

X. Acknowledgments

This work was partially supported by the EU FP7 T-NOVA Project (619520) and the Collaborative Research Center 1053 (MAKI) funded by the German Research Foundation (DFG).

References

[1] 4WARD Project, http://www.4ward-project.eu/.
[2] Amazon EC2 Instance Types, http://aws.amazon.com/ec2/instance-types/.
[3] ETSI Network Function Virtualization, http://www.etsi.org/technologies-clusters/technologies/nfv
[4] GENI: Global Environment for Network Innovations, http://www.geni.net/.
[5] OPNFV, https://www.opnfv.org/
[6] SONATA Project, http://www.sonata-nfv.eu/
[7] T-NOVA Project, http://www.t-nova.eu/.
[8] UNIFY Project, http://www.fp7-unify.eu/
[9] A. Abujoda and P. Papadimitriou, MIDAS: Middlebox Discovery and Selection for On-Path Flow Processing, IEEE COMSNETS, Bangalore, India, January 2015.
[10] A. Abujoda and P. Papadimitriou, DistNSE: Distributed Network Service Embedding Across Multiple Providers, IEEE COMSNETS, Bangalore, India, January 2016.
[11] D. Adami et al., Effective resource control strategies using OpenFlow in cloud data center, IFIP/IEEE International Symposium on Integrated Network Management, 2013.
[12] B. Aggarwal et al., EndRE: An end-system redundancy elimination service for enterprises, ACM NSDI 2010, San Jose, USA, April 2010.
[13] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, Towards predictable datacenter networks, ACM SIGCOMM 2011, Toronto, Canada, August 2011.
[14] M. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, Orchestrating Virtualized Network Functions in NFV, 2015.
[15] M. T. Beck and J. F. Botero, Coordinated Allocation of Service Function Chains, IEEE GLOBECOM, December 2015.
[16] T. Benson, A. Akella, A. Shaikh, and S. Sahu, CloudNaaS: a cloud networking platform for enterprise applications, ACM SOCC 2011, Cascais, Portugal, October 2011.
[17] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, Queueing Networks and Markov Chains, Wiley, 2006.
[18] M. Chowdhury , M. Rahman, and R. Boutaba, Virtual Network Embedding with Coordinated Node and Link Mapping, IEEE Infocom 2009, Rio de Janeiro, Brazil, April 2009.
[19] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, Near Optimal Placement of Virtual Network Functions, IEEE INFOCOM, Hong Kong, China, April 2015.
[20] D. Dietrich, A. Abujoda, and P. Papadimitriou, Network Service Embedding Across Multiple Providers with Nestor, IFIP Networking, Toulouse, France, May 2015.
[21] D. Dietrich, A. Rizk, and P. Papadimitriou, Multi-Provider Virtual Network Embedding with Limited Information Disclosure, IEEE Transactions on Network and Service Management, 12(2), June 2015, pp.188–201.

[22] M. Dobrescu, K. Argyarki, and S. Ratnasamy, Toward Predictable Performance in Software Packet-Processing Platforms, USENIX NSDI, San Jose, USA, April 2012.
[23] M. Dobrescu et al., RouteBricks: exploiting parallelism to scale software routers, ACM SOSP, Big Sky, USA, October 2009.
[24] S. Fayazbakhsh et al., Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags, USENIX NSDI 2014, Seattle, USA, April 2014.
[25] B. Frank et al., Pushing CDN-ISP Collaboration to the Limit, ACM SIGCOMM CCR, 43(3), 2013.
[26] A. Gember et al., Stratos: Virtual Middleboxes as First-Class Entities, Technical Report TR1771, 2012.
[27] R. Guerzoni et al., A novel approach to virtual networks embedding for SDN management and orchestration, IEEE Network Operations and Management Symposium (NOMS) 2014.
[28] S. Guo et al., SecondNet: a data center network virtualization architecture with bandwidth guarantees, ACM CONEXT 2010, New York, USA, December 2010.
[29] I. Houidi et al., Virtual Network Provisioning Across Multiple Substrate Networks, Computer Networks, 55(4), March 2011.
[30] J. Lee et al, CloudMirror: Application-Aware Bandwidth Reservations in the Cloud USENIX HotCloud, San Jose, USA, June 2013.
[31] M. C. Luizelli et. al, Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions, IFIP/IEEE IM, May 2015.
[32] T. Lukovszki, M. Rost, and S. Schmid, It's a Match!: Near-Optimal and Incremental Middlebox Deployment, ACM SIGCOMM CCR, 46(1), Jan. 2016, pp. 30–36.
[33] T. Lukovszki and S. Schmid, Online admission control and embedding of service chains, SIROCCO, Montserrat, Spain, July 2015.
[34] B. Martini et al., Latency-aware composition of Virtual Functions in 5G, IEEE Conference on Network Softwarization (NetSoft), London, 2015
[35] S. Mehraghdam, M. Keller, and H. Karl, Specifying and placing chains of virtual network functions, IEEE CloudNet, Luxembourg, October 2014
[36] Z. Qazi et al, SIMPLE-fying middlebox policy enforcement using SDN ACM SIGCOMM 2013, Hong Kong, China, August 2013.
[37] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, Network service chaining with optimized network function embedding supporting service decompositions, Computer Networks, 93(3), December 2015, pp. 492-505.
[38] G. Schaffrath et al., Network Virtualization Architecture: Proposal and Initial Prototype, ACM SIGCOMM VISA, Barcelona, Spain, August 2009.
[39] V. Sekar, N. Egi, S. Ratnasamy, M. Reiter, and G. Shi, The Design and Implementation of a Consolidated Middlebox Architecture, USENIX NSDI, San Jose, USA, April 2012.
[40] J. Sherry et al., Making Middleboxes Someone Elses Problem: Network Processing as a Cloud Service, ACM SIGCOMM 2012, Helsinki, Finland, August 2012.
[41] B. Sonkoly et al., Multi-Domain Service Orchestration Over Networks and Clouds: A Unified Approach, SIGCOMM Comput. Commun. Rev., 45(4), August 2015, pp. 377–378.
[42] N. Spring, R. Mahajan, and D. Wetherall, Measuring ISP Topologies with RocketFuel, ACM SIGCOMM, Pittsburgh, USA, August 2002.
[43] N. Spring and D. Wetherall, A Protocol Independent Technique for Eliminating Redundant Network Traffic, ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.
[44] C. Xenakis et al., A generic Characterization of the Overheads Imposed by IPSec and Associated Cryptographic Algorithms, Computer Networks, 50(17), December 2006, pp. 3225–3241.
[45] Q. Wu and T. Wolf, Runtime Task Allocation in Multi-Core Packet Processing Systems, IEEE Transactions on Parallel and Distributed Systems, 23(10), October 2012, pp. 1934–1943.
[46] M. Yu et al., Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration, ACM SIGCOMM Computer Communications Review, 38(2), April 2008, pp. 17–29.
[47] Y. Zu et al., Cabernet: Connectivity Architecture for Better Network Services, ACM ReArch '08, Madrid, Spain, December 2008.
[48] Q. Zhang et al., Vertex-centric computation of service function chains in multi-domain networks, IEEE NetSoft, June 2016.
[49] Y. Zhu and M. Ammar, Algorithms for Assigning Substrate Network Resources to Virtual Network Components, IEEE INFOCOM, Barcelona, Spain, April 2006.

David Dietrich received the doctoral degree (Dr.-Ing.) from the Leibniz Universität Hannover, Germany, in 2016 and the M.Sc. from the Universität Kassel, Germany, in 2006. During the years 2006 until 2009 he was a software engineer for real-time simulation of automotive communications networks. In 2010, he joined the Institute of Communications Technology at the Leibniz Universität Hannover where he works as postdoctoral researcher since 2016. He also participated in several research projects, *e.g.,* the EU-funded T-NOVA project and the national project G-Lab. His research interests include network virtualization and network management with focus on algorithms and optimization. David Dietrich is a member of IEEE and ACM.

Ahmed Abujoda is a research associate at the Institute of Communications Technology at the Leibniz Universität Hannover, Germany, where he also received his PhD in electrical and computer engineering. He graduated in electrical engineering from the university of Sana'a, Yemen, in 2004. He also obtained a master of science degree in Mechatronics engineering from Universität Siegen, Germany. He worked in EU-funded projects such as T-NOVA and CONFINE. His research interests include software-defined networking, network function virtualization and network management.

Amr Rizk received his doctoral degree (Dr.-Ing.) from the Leibniz Universität Hannover, Germany, in 2013. He has been a Post-Doctoral research fellow with the Department of Computer Science at the University of Warwick, UK in 2014 and with the University of Massachusetts (UMass), Amherst in 2015. Currently, he is a Post-Doctoral research group head at the Multimedia Communications Lab at the TU Darmstadt, Germany. Amr received a stipend from the German Academic Exchange Service for his research stay at UMass in 2015. He also received the excellence in DASH Award at ACM MMSyst 2016. His research interests are in the areas of performance evaluation of communication networks, stochastic modeling, software-defined networks and teletraffic theory.

Panagiotis Papadimitriou is faculty at the department of Applied Informatics in the University of Macedonia, Greece, and a member of the L3S Research Center in Hannover, Germany. During 2011–2016, he was an Assistant Professor at the Communications Technology Institute of Leibniz Universität Hannover, Germany. Panagiotis received a Ph.D. in Electrical and Computer Engineering from Democritus University of Thrace, Greece, in 2008. He has participated in several EU-funded (*e.g.,* T-NOVA, CONFINE, 4WARD, SAIL) and national projects (*e.g.,* EPSRC VROUTER, G-Lab), and received the runner-up Poster Award at ACM SIGCOMM 2009 and the Best Paper Award at IFIP WWIC 2012. Panagiotis has been the TPC co-chair of international conference and workshops, such as IEEE INFOCOM SWFAN 2016 and IFIP WWIC 2016. His research activities currently span next-generation Internet architectures, NFV, SDN, cloud networking, and mobile edge computing. Panagiotis is a Senior Member of IEEE and a member of ACM.