Network Service Embedding Across Multiple Providers with Nestor

David Dietrich Ahmed Abujoda Panagiotis Papadimitriou
Institute of Communications Technology, Leibniz Universität Hannover, Germany
{*first.last*}@*ikt.uni-hannover.de*

***Abstract*—The migration of network functions (NFs) into virtualized network infrastructures brings significant benefits to enterprise networks, while creating opportunities for new cloud service models (i.e., NF-as-a-Service). Network service embedding (NSE) entails serious challenges, stemming from middlebox policies prescribed by network operators and the implications of NFs on network traffic (*i.e.,* bandwidth conservation or traffic amplification) that complicate the estimation of bandwidth demands. The NSE problem is further exacerbated by the location dependencies of certain NFs, which, in conjunction with the limited geographic footprint of NF providers, raise the need for network service mapping across multiple providers.**

In this paper, we present a holistic approach to multi-provider NSE. We introduce a new service model that simplifies the specification of network service requests and the estimation of bandwidth demands. We further define topology abstractions tailored to NSE that are exposed to a network service composition layer (NSCL), interposed between the clients and the NF providers. Based on this service model and topology abstractions, we propose Nestor, a system that generates efficient network service embeddings via network graph rendering, request partitioning among datacenters (DCs), and request segment mappings onto DC networks.

I. Introduction

Middleboxes embed a wide range of flow processing functions into the network infrastructure, satisfying the increasing needs of network operators and end-users. Despite their widespread adoption, middleboxes exhibit significant limitations in terms of customization, resource efficiency, and manageability [21], [22]. More precisely, middleboxes are typically built of specialized hardware and offer single-purpose functionality, leading to appliance sprawl, and increased capital and operational expenses for enterprise networks.

Network function virtualization (NFV) is an emerging concept that aims at mitigating some of these problems by enabling the consolidation of network functions (NF) on platforms built of commodity components [2], [3]. This can reduce the expenses for NF deployment either by deploying consolidated software middleboxes in enterprise networks or by outsourcing NFs to virtualized network infrastructures. The latter, in particular, is very appealing to enterprises, since NF-as-a-service (NFaaS) obviates the need to acquire, deploy, and operate additional network appliances on clients' premises, leading to significant savings in operational and technology investment costs. Furthermore, the recent trend of micro-datacenter deployment by large Internet Service Providers (ISPs) [13] leads to a larger number of NFV Points-of-Presence (PoP) and essentially better NFaaS offerings for clients.

ISBN 978-3-901882-68-5

Middlebox deployment is commonly driven by certain policies, *i.e.,* traffic should traverse a set of middleboxes in a specific order. In enterprise networks, such policies are enforced with careful middlebox interposition. Middlebox policies should be preserved when NFs are migrated to a virtualized infrastructure. Ideally, NF outsourcing should be indistinguishable from on-site middlebox deployment in terms of policy. This entails significant challenges for network service embedding (NSE) and provisioning. While recent work has addressed some of the network service provisioning issues (*e.g.,* packet header modifications hindering policy enforcement) by employing flow tagging [12] or NF-graph transformations [14], [7], existing solutions for NSE are limited to heuristic algorithms that primarily aim at minimizing inter-rack traffic in data-centers (DC) [7], [14].

The 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 NF Providers (NFP). More precisely, a single NFP may not satisfy the location constraints of all NFs in a 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 network providers to third parties) [9]. Existing solutions for multi-provider virtual network (VN) mapping [9], [16] generate embeddings based on VN graphs with specific bandwidth demands on each edge. This highly abstract service model cannot represent network service compositions and the NF implications on traffic (*i.e.,* certain NFs reduce or amplify traffic).

In this paper, we present a holistic approach to multi-provider NSE. In particular, we propose a NSE orchestrator, called *Nestor*, which generates efficient embeddings via network graph rendering, request partitioning among DCs, and request segment mappings onto the DC networks. In this respect, we introduce a new service model, tailored to NSE,

that simplifies (i) the specification of network service requests and (ii) the estimation of computing and bandwidth demands for each request. We further define a topology abstraction that facilitates request partitioning while obscuring any information that is deemed confidential by NFPs. Nestor provides the rendering of such topology abstractions from detailed topology graphs accessed only by NFPs. We decouple network service composition from NFPs by interposing a network service composition layer (NSCL) between the clients and the NFPs. The NSCL uses two variants of an integer linear program (tailored to the client or the NFP) to assign NFs to DCs and subsequently generates NF-subgraphs (*i.e.,* request segments) mappable to DC networks, by employing virtual gateways for inter-segment traffic aggregation. Upon request partitioning, each NF-subgraph is embedded onto the corresponding DC, aided by the binding of the subgraph's virtual gateway with the DC network gateway. To this end, we present a heuristic algorithm and a mixed-integer program, executed by each NFP.

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

II. Service and Topology Abstractions

Service chaining is a common abstraction for the expression of network service requirements [14], [19]. A service chain represents the exact sequence of NFs traversed by one or multiple flows. Fig. 1 illustrates an example of service chaining. In particular, 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 traverse 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.

NSE consists in mapping service chains across multiple DCs. We particularly consider DCs operated by multiple NFPs, since the footprint of individual NFPs may not satisfy the location dependencies of all NFs in a chain. NSE requests are formulated by the client (*e.g.,* an enterprise). Besides the NFs that compose the service, a NSE request is associated with bandwidth demands between pairs of NFs. NSE should generate a service mapping, such that NF location dependencies and resource requirements (*i.e.,* CPU and bandwidth) are satisfied.

Since NFP policies will hinder interoperability with third parties and especially competitors, we interpose a network service composition layer (NSCL) between the clients and the NFPs. NSCL is delegated with the assignment of NFs to DCs operated by multiple NFPs. This is a task that NSCL can carry out with an abstract network view (Section II-B).

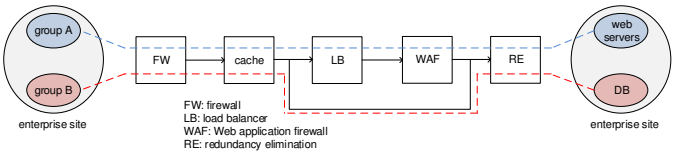


Fig. 1. Service chaining example.

On the contrary, the assignment of NFs to servers requires detailed knowledge of the network topology and resource availability, and as such, it can be performed only by the NFPs. These observations lead to the NSE problem decomposition that Nestor employs. Before elaborating on our NSE approach (Section III), we discuss two critical aspects of NSE: (i) a service model that circumvents the difficulty in specifying bandwidth demands in a service chain and (ii) topology abstractions that facilitate NSE while adhering to NFPs' information disclosure policies.

A. Service Model

We aim at defining a service model that simplifies the specification of service requirements by clients and the estimation of NF computing and bandwidth demands. The difficulty in computing resource requirements stems from the effects of NFs on traffic. More precisely, appliances, such as REs and caches, conserve bandwidth, while other NFs (*e.g.,* packet multiplication, encryption) amplify traffic. The level of bandwidth conservation or traffic amplification depends on various factors, such as the size and hit ratio for caches, the amount of duplicate content for RE appliances, and the volume of traffic filtered by firewalls and intrusion detection systems (IDS). In this respect, Table I summarizes the effect of widely-used NFs on traffic and further shows the range of bandwidth saving or traffic amplification for each NF, collected from various studies [24], [5], [25]. Based on these observations, we introduce φ_p^i which denotes the ratio of outbound traffic at port p of NF i over the aggregate inbound traffic at all ports. We particularly consider the traffic ratio per output port, since traffic may be split between multiple output ports depending on the outcome of packet inspection.

Our network service model consists of a NF-graph in which each NF i is associated with a traffic ratio φ_p^i per port p (Fig. 2). 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 given 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 [24], [5], [25] 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 bandwidth saving or the highest level of traffic amplification (assuming a known range of bandwidth saving or traffic amplification, 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

TABLE I
EFFECTS OF NETWORK FUNCTIONS ON TRAFFIC RATE.

Network function	Bandwidth preservation	Outbound/inbound traffic rate (φ)
Flow monitoring	Yes	-
Load balancer	Yes	-
NAT	Yes	-
RE	No	40–70% [24], 59–74% [5]
VPN (IPsec)	No	105–228% (for 64–1500-byte packets) [25]

chain, the client simply needs to specify the rate of the traffic generated at each end-point.

The computational requirements for each NF can be derived using 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 [11], [10], while existing profiling techniques [26] can be applied to any flow processing workloads whose computational requirements are not known. This obviates the need to specify any computing demands for the NFs in the service chain.

B. Topology Abstractions

Topology abstractions are crucial for the generation of efficient embeddings considering the information disclosure policies of NFPs. We seek to identify topology abstractions that conceal any information deemed as confidential by NFPs. To this end, we rely on information disclosed by ISPs and cloud providers. ISPs often publish simplified PoP-level topologies [23], while cloud providers advertise resource types across different availability zones [1].

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. 3(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. 3(a), we derive an abstract network view that obscures the Internet access points and represents (i) the connectivity between DCs and peering nodes within each NFP, and (ii) the peerings among NFPs (Fig. 3(b)). This topology abstraction combined with NF computing and link costs provides the necessary means for the estimation of the overall NSE cost.

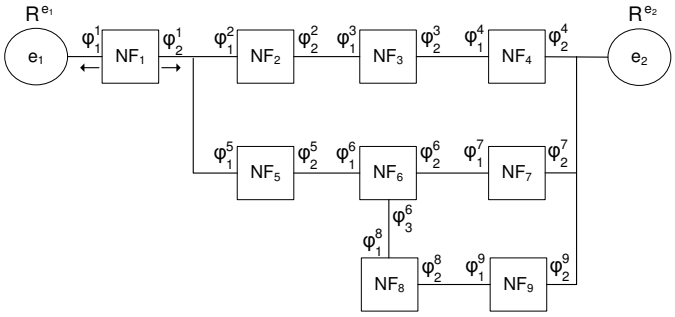
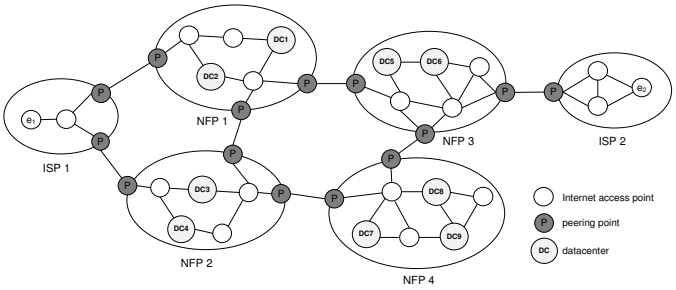
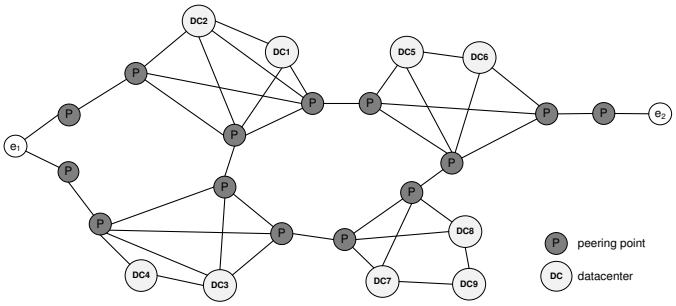


Fig. 2. Service model (R represents the traffic rate generated at an end-point and φ_p^i denotes the outbound to inbound traffic ratio at the port p of the NF i).



(a) PoP-level topology



(b) Topology abstraction

Fig. 3. Topology abstraction for request partitioning.

The edges of this network 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 request partitioning formulation variant which is tailored to NFPs (Section IV-B).

III. NESTOR OVERVIEW

In this section, we provide an overview of Nestor and discuss the sequence of steps for the embedding of network service requests. Nestor processes and embeds requests specified based on the service model presented in Section II-A. The topology abstraction in Section II-B represents the view of the NSCL on the substrate network topologies. To embed network service requests, Nestor implements a NSE control plane, which is distributed across the NSCL, the NFPs, and the DCs deployed by each NFP, as shown in Fig. 4. Along these lines, Nestor decomposes NSE into the following steps:

Graph Rendering. Graph rendering consists in the transformation of detailed topology graphs into topology abstractions that facilitate request 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 (*i.e.,* expressed as cost per bandwidth unit) and optionally with

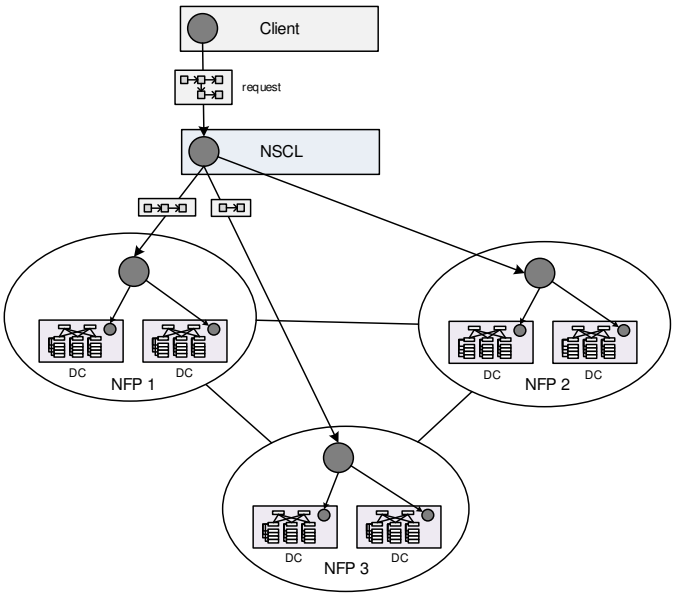
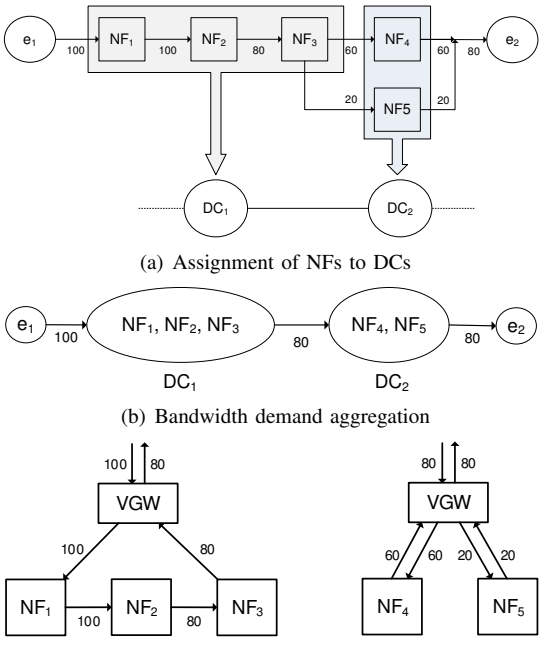


Fig. 4. Nestor control plane overview.



(a) Assignment of NFs to DCs

(b) Bandwidth demand aggregation

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

Fig. 5. Request partitioning.

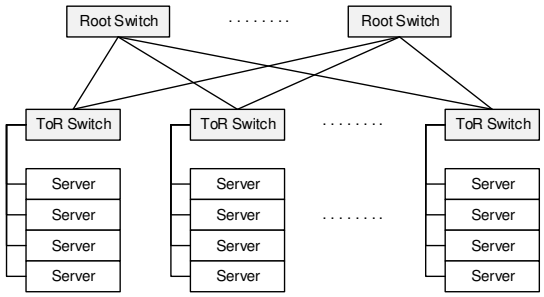


Fig. 6. DC network topology.

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. 3(b)). New topology abstractions are generated upon significant substrate topology changes or the participation of new NFPs. Link weights are updated on the existing network graphs in response to changes in resource utilization levels or NFP policies.

Request Partitioning. Network service requests are partitioned among NFPs, when there is no single NFP that satisfies the location dependencies of all NFs in the request. More precisely, the NSCL identifies a list of DC candidates for each requested NF by matching NF location constraints against each NFP's footprint. Subsequently, the NSCL uses two variants of an integer linear program for request partitioning, tailored to (i) the client (*i.e.,* expenditure minimization) or (ii) the NFPs (*i.e.,* network load balancing), exploiting link and DC preferences disclosed by NFPs. The request partitioning formulations are discussed in detail in Section IV-B.

The request segments are derived from the ILP solver output, *i.e.,* the NF-to-DC assignment (Fig. 5(a)). First, the NSCL computes the total inbound and outbound bandwidth demand for each request segment (Fig. 5(b)). Next, the NSCL generates a NF-subgraph, in which all inter-segment traffic traverses a virtual gateway (VGW), as shown in Fig. 5(c). This subgraph allows the binding of the VGW with the DC network gateway, augmenting the mapping of each request segment 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 (Fig. 6) 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 two methods: (i) a mixed integer program formulation for exact solutions and (ii) a heuristic algorithm for approximate but fast retrievable solutions. The objective of both methods is the maximization of NF consolidation level and minimization of inter-rack traffic. Further details on these NF-subgraph mapping methods are given in Sections IV-C and IV-D.

IV. Network Service Embedding Methods

In this section, we present our methods for request partitioning and NF-subgraph mapping. We first introduce our request and network model (Section IV-A). Next, we present an integer linear program (ILP) formulation for request partitioning (Section IV-B). We also define a mixed integer program (MIP) formulation (Section IV-C) and a heuristic algorithm for the mapping of NF-subgraphs onto DC networks (Section IV-D).

Furthermore, we compare the MIP and the heuristic in terms of NF-subgraph mapping efficiency (Section IV-E).

A. Request and Network Model

We briefly discuss our request and network model. Both stem from the service model and topology abstractions presented in Section II.

Request Model. We use a directed graph $G_F = (V_F, E_F)$ to express a network service request. The set of vertices V_F includes all NFs and end-points that comprise the service request. Each NF i is associated with an outbound/inbound traffic ratio 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 rely on an undirected graph $G_S = (V_S, E_S)$ for the description of topology abstractions (Section II-B) and substrate network topologies. We use α_u and β_{uv} to express the monetary cost of NFs and links, respectively. As discussed in Section II-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, 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. 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 $Mbps$
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 $Mbps$
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

B. Request Partitioning

Request partitioning will be subject to objectives, such as service cost minimization or network load balancing. In this respect, we provide two ILP formulation variants, tailored to the client and the NFPs. The ILP variants differ only in the objective function. The objective function **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 the link weights, disclosed by NFPs. Since link weights express the utilization of links and DCs, link weight minimization essentially leads to network load balancing.

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 request partitioning ILP is defined as follows:

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:

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

$$\sum_{\substack{v \in V_S \\ (u \neq v)}} (y_{uv}^{ij} - y_{vu}^{ij}) = x_u^i - x_u^j \qquad i \neq j, \forall (i,j) \in E_F, \forall u \in V_S \tag{4}$$

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

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

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

Hereby, we briefly discuss the ILP constraints. Constraint (3) ensures that each NF i 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}. In addition, we fix the assignment of each end-point k in the request to its respective location u by setting $x_u^k \leftarrow 1$.

We rely on the branch-and-cut method for solving the ILPs. The request partitioning ILP solver yields a mean runtime of 210 ms (with the evaluation parameters shown in Table III). Time complexity and solver runtime can be reduced by

employing relaxation and rounding techniques at the cost of suboptimality [8].

C. NF-subgraph Mapping MIP

The MIP for NF-subgraph mapping aims at maximizing NF co-location while minimizing the traffic within the DC. In this respect, the binary variable x_u^i denotes the assignment of NF i to the server u, while the binary variable z_u indicates whether the server u has been assigned to any NFs (*i.e.,* $z_u = 0$ when there is no NF assigned to server u; otherwise $z_u = 1$). Based on the multi-commodity flow problem formulation, we use the term *commodity*, defined as $Com^{ij} = \{i, j, d^{ij}\}$, to express bandwidth demands d^{ij} between a pair of NFs i, j. In this context, the flow variable f_{uv}^{ij} denotes the amount of flow (*i.e.,* bandwidth units) over the DC link (u, v) for the NF-graph edge $(i, j) \in E_F$.

The NF-subgraph mapping MIP is formulated as follows:

Minimize

$$\sum_{u \in V_S} z_u + \frac{1}{\sum\limits_{(i,j) \in E_F} d^{ij}} \cdot \sum_{\substack{(u,v) \in E_S \\ (u \neq v)}} \sum_{(i,j) \in E_F} f_{uv}^{ij} \tag{8}$$

subject to:

$$\sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_F \tag{9}$$

$$\sum_{\substack{v \in V_S \\ (u \neq v)}} (f_{uv}^{ij} - f_{vu}^{ij}) = d^{ij}(x_u^i - x_u^j)$$
$$i \neq j, \forall (i,j) \in E_F, \forall u \in V_S \tag{10}$$

$$\sum_{i \in V_F} d^i x_u^i \leq r_u z_u \quad \forall u \in V_S \tag{11}$$

$$\sum_{i,j \in V_F} f_{uv}^{ij} \leq r_{uv} \quad \forall u, v \in V_S \tag{12}$$

$$x_u^i, z_u \in \{0, 1\} \quad \forall i \in V_F, \forall u \in V_S \tag{13}$$

$$f_{uv}^{ij} \geq 0 \quad \forall (i,j) \in E_F, \forall (u,v) \in E_S \tag{14}$$

The objective function (8) consists of two terms, *i.e.,* the number of assigned servers and the accumulated flow divided by the total bandwidth demand. Essentially, the second term yields 1 if all NF-graph edges $(i, j) \in E_F$ are mapped onto single-hop links. The normalization of the second term provides a balance against the first term in the objective function.

We further discuss the constraints (9)–(14) in the MIP. Condition (9) ensures that each NF $i \in V_F$ is mapped exactly to one server. Constraint (10) enforces flow conservation, *i.e.,* the sum of all inbound and outbound traffic in switches and servers that do not host NFs should be zero. The constraints (11) and (12) ensure that the allocated computing and bandwidth resources do not exceed the residual capacities of servers and links, respectively. Finally, condition (13) expresses the binary domain constraint for the variables x_u^i and z_u, while constraint (14) 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 $x_{GW}^{V_F(1)} \leftarrow 1$.

D. NF-subgraph Mapping Heuristic

Similar to the MIP, the NF-subgraph mapping heuristic strives to maximize NF co-location and minimize inter-rack traffic. To this end, the algorithm sorts (i) the racks in decreasing order, according to the available bandwidth between their top-of-the-rack (ToR) switch and the root switches, and (ii) the servers in each rack in decreasing order, according to their residual computing capacity. Subsequently, the algorithm examines the feasibility of assigning the NF-subgraph into any of the racks, taking into account the residual computing and link capacity within each rack. If this is not feasible, the algorithm splits the NF-subgraph into two segments using *min-cut*. This ensures that the most heavily communicating components are assigned to the same segment, reducing the traffic between the segments and, thereby, between the racks. For any of the segments that do not fit into any of the racks, the algorithm performs additional iterations partitioning segments, till all segments have been assigned; otherwise, the request is rejected.

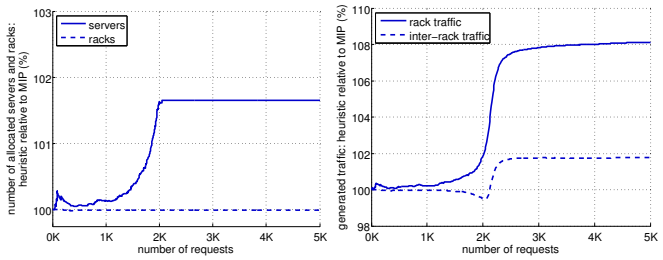
E. NF-subgraph Mapping Efficiency

We hereby investigate the suboptimality of the heuristic compared to the MIP, in terms of DC mapping efficiency. To this end, we assign 5000 non-expiring NF-graphs onto a DC, using both mapping variants. Each NF-graph contains 2 to 20 NFs. The evaluation parameters used for this test are the same with our NSE evaluation in Section V (see Table III).

We aim at comparing the level of NF consolidation and inter-rack traffic of both NF-graph mapping variants. In this respect, Fig. 7(a) illustrates the relative utilization of DC servers and racks. The heuristic results in marginally higher server utilization, while rack utilization remains the same for both DC mapping methods. In the long run, both variants achieve a CPU utilization of around 92.5%. This indicates that NFPs can generate equally high revenues from NFaaS offerings.

We further investigate whether the heuristic generates additional traffic within the DC, compared to the MIP. According to Fig. 7(b), the heuristic results in marginally higher volume of inter-rack traffic and a more perceptible increase (*i.e.,* up to 8%) in the traffic within the racks. This indicates nearly no suboptimality in terms of inter-rack traffic minimization for the heuristic. Since inter-rack traffic is associated with NF-graph partitioning among racks, we infer that the heuristic has a higher tendency to partition NF-graphs within the same rack rather than across racks. This justifies the marginally higher utilization of servers by the heuristic, as illustrated in Fig. 7(a).

Eventually, the heuristic yields only marginal suboptimality compared to the MIP. This mainly stems from a slightly lower NF consolidation level. However, the heuristic exhibits



(a) Server and rack allocation (b) DC traffic generation

Fig. 7. NF-graph mapping with MIP and heuristic algorithm.

a substantially lower runtime (4 ms) compared to the MIP (80 ms)[1], which we deem that outweighs its suboptimality. As such, we employ the heuristic into Nestor for NF-subgraph mapping in our NSE evaluation study (Section V).

V. Evaluation

In this section, we assess the efficiency of NSE across multiple NFPs with Nestor. We mainly focus on request 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 request partitioning ILP variants, introduced in Section IV-B. Upon partitioning, the mapping of NF-subgraphs to DCs is generated with the heuristic algorithm presented in Section IV-D. In the following, we present our evaluation environment (Section V-A) and discuss our evaluation results (Section V-B).

A. Evaluation Environment

We have implemented an evaluation environment for multi-provider NSE in C/C++. Our implementation includes the Nestor NSE framework (Section III), a service chain generator, and a substrate network topology generator. We rely on CPLEX as optimizer for our ILP/MIP. 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 with the size of the US state 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, similar to Fig. 6.

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 the outbound/inbound traffic rate (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 II-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. Table III provides a list of the evaluation parameters.

[1]Tests are carried out on a server with two quad-core Intel Xeon CPUs at 2.53 GHz.

TABLE III
Evaluation Parameters.

Substrate network (PoP-level topology):	
NFPs / DCs	12 / 50
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
Substrate network (Data Center topology):	
Root switches	5
Racks per DC	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

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.
- **Acceptance rate** is the number of successfully embedded requests over the total number of requests.
- **Revenue** accumulates the CPU and bandwidth units leased to clients.

B. Evaluation Results

We perform a comparative study between the two request partitioning ILP variants (Section IV-B), *i.e.,* embedding cost 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 or the 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.

Fig. 8 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, compared to *Min-C* which is formulated 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. 9 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

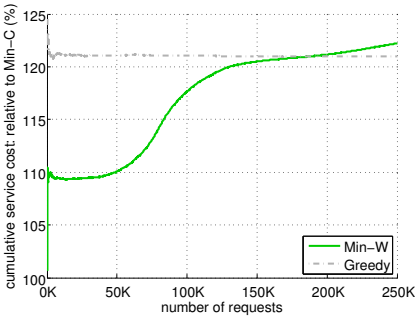


Fig. 8. Cumulative service cost with weight-minimized and greedy request partitioning relative to the cost-minimized solution.

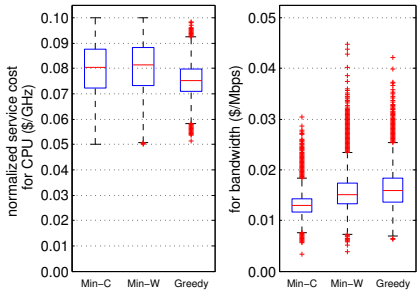


Fig. 9. Normalized service cost with cost-minimized, weight-minimized, and greedy request partitioning.

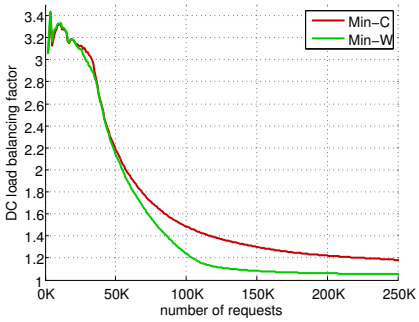


Fig. 10. Load balancing level with cost- and weight-minimized request partitioning.

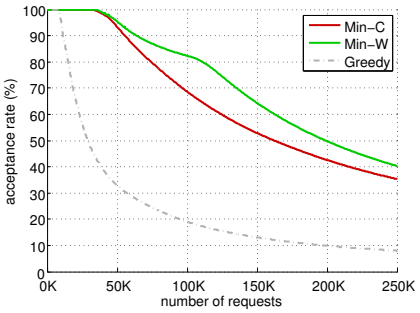


Fig. 11. Acceptance rate with cost-minimized, weight-minimized, and greedy request partitioning.

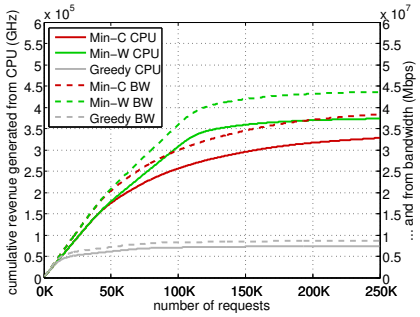


Fig. 12. Cumulative revenue with cost-minimized, weight-minimized, and greedy request partitioning.

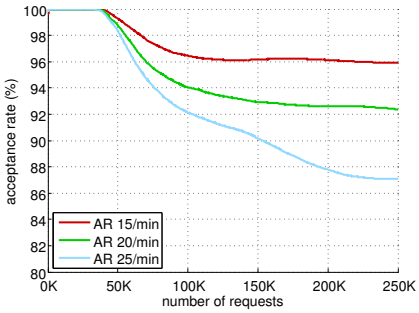


Fig. 13. Acceptance rate with diverse arrival rates of expiring requests and weight-minimized request partitioning.

the significantly lower bandwidth cost (Fig. 9), 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. 10 depicts the evolution of load balancing level across the DCs. Since the greedy selection of DCs close to the end-points does not lead to load balancing, we focus on the load balancing levels of the two ILP variants. According to Fig. 10, *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 a perceptible suboptimality. 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.

Fig. 11 shows the request acceptance rates for the three request 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.

Figs. 11 and 12 show a strong correlation between the acceptance rate and generated revenue. The ILP 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 request partitioning method for NFPs.

Finally, we measure the acceptance rate of *Min-W* with 250K expiring requests and diverse arrival rates. Fig. 13 shows that acceptance rates converge to a steady state, irrespective of the arrival rate. This further indicates the efficiency of *Min-W* ILP and our DC mapping algorithm for NSE across multiple NFPs.

VI. Related Work

We briefly discuss related work on NF-graph mapping and request partitioning for network service and VN embedding.

NF-graph mapping. Existing work on NSE has mainly focused on NF-graph mapping onto DC networks. STRATOS [14] and CloudNaaS [7] propose heuristic mapping algorithms that seek to minimize inter-rack traffic within DC networks. A similar approach is also taken by Oktopus [6], SecondNet [15] and CloudMirror [18] for the assignment of virtual clusters to DCs. *Huan et al.* propose a distributed algorithm for network service placement assuming the ability to deploy NFs in the

data path [17]. MIDAS [4] employs a heuristic algorithm for order-preserving NF assignment to middleboxes deployed along the data path. Several studies have tackled the problem of embedding VN topologies onto a shared substrate network, relying on heuristic algorithms [27], [28] or linear programs [8], [9], [16]. However, these embedding techniques are designed for arbitrary virtual and substrate network topologies, and are not optimized for NF-graph mapping onto DC network topologies. Compared to all these approaches, Nestor provides a holistic solution for the NSE problem across multiple NFPs, including request partitioning, generation of NF-subgraphs with DC gateway bindings, and NF-subgraph mapping.

Request partitioning. Authors in [16], [9] apply a similar VN embedding problem decomposition, *i.e.,* VN request partitioning among providers, followed by request segment mapping onto each provider's network. However, both frameworks [16], [9] pertain only to VN embedding, as their request models cannot represent service chaining while request partitioning has been formulated assuming only the knowledge of provider peerings. In contrast, Nestor couples service chaining with NF traffic ratios to simplify the estimation of bandwidth demands, uses a topology abstraction tailored to NSE, and optimizes request partitioning with respect to the client's expenditure and NFPs' policies.

VII. Conclusions

In this paper, we presented Nestor, a NSE orchestrator that addresses the main challenges faced by the assignment of service chains across multiple NFPs. Nestor encompasses a service model that simplifies the estimation of computational and bandwidth requirements in service chains, and topology abstractions that augment request partitioning, while obscuring any confidential information for NFPs. Nestor relies on a broker (*i.e.,* NSCL) for request partitioning and the generation of NF-subgraphs mappable to DC networks. We presented request partitioning ILP variants optimized for the client and the NFPs. The mapping of NF-subgraphs to DCs is computed by a heuristic algorithm which yields marginal suboptimality compared to a MIP.

Our evaluation results show the high efficiency of Nestor with both request partitioning ILP variants. Our insights into request partitioning uncover a trade-off between service cost minimization and resource efficiency. 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.

VIII. Acknowledgments

We thank Amr Rizk for his comments on the paper. This work was partially supported by the EU FP7 T-NOVA Project (619520).

References

[1] Amazon EC2 Instance Types, http://aws.amazon.com/ec2/instance-types/.
[2] ETSI Network Function Virtualization, http://www.etsi.org/technologies-clusters/technologies/nfv
[3] T-NOVA Project, http://www.t-nova.eu/.
[4] A. Abujoda and P. Papadimitriou, MIDAS: Middlebox Discovery and Selection for On-Path Flow Processing, IEEE COMSNETS, Bangalore, India, January 2015.
[5] B. Aggarwal et al., EndRE: An end-system redundancy elimination service for enterprises, ACM NSDI 2010, San Jose, USA, April 2010.
[6] H. Ballani et al., Towards predictable datacenter networks ACM SIGCOMM 2011, Toronto, Canada, August 2011.
[7] T. Benson et al., CloudNaaS: a cloud networking platform for enterprise applications, ACM SOCC 2011, Cascais, Portugal, October 2011.
[8] 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.
[9] D. Dietrich, A. Rizk, and P. Papadimitriou, Multi-Domain Virtual Network Embedding with Limited Information Disclosure, IFIP Networking, New York, USA, May 2013.
[10] M. Dobrescu, K. Argyarki, and S. Ratnasamy, Toward Predictable Performance in Software Packet-Processing Platforms, USENIX NSDI, San Jose, USA, April 2012.
[11] M. Dobrescu et al., RouteBricks: exploiting parallelism to scale software routers, ACM SOSP, Big Sky, USA, October 2009.
[12] S. Fayazbakhsh et al., Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags, USENIX NSDI 2014, Seattle, USA, April 2014.
[13] B. Frank et al., Pushing CDN-ISP Collaboration to the Limit, ACM SIGCOMM CCR, 43(3), 2013.
[14] A. Gember et al., Stratos: Virtual Middleboxes as First-Class Entities, Technical Report TR1771, 2012.
[15] S. Guo et al., SecondNet: a data center network virtualization architecture with bandwidth guarantees, ACM CONEXT 2010, New York, USA, December 2010.
[16] I. Houidi et al., Virtual Network Provisioning Across Multiple Substrate Networks, Computer Networks, 55(4), March 2011.
[17] X. Huang, S. Ganapathy, and T. Wolf, A Scalable Distributed Routing Protocol for Networks with Data-Path Services, IEEE INCP 2008, Orlando, Florida, USA, October 2008.
[18] J. Lee et al, CloudMirror: Application-Aware Bandwidth Reservations in the Cloud USENIX HotCloud, San Jose, USA, June 2013.
[19] Z. Qazi et al, SIMPLE-fying middlebox policy enforcement using SDN ACM SIGCOMM 2013, Hong Kong, China, August 2013.
[20] G. Schaffrath et al., Network Virtualization Architecture: Proposal and Initial Prototype, ACM SIGCOMM VISA, Barcelona, Spain, August 2009.
[21] V. Sekar et al.,The Design and Implementation of a Consolidated Middlebox Architecture, USENIX NSDI, San Jose, USA, April 2012.
[22] J. Sherry et al., Making Middleboxes Someone Elses Problem: Network Processing as a Cloud Service, ACM SIGCOMM 2012, Helsinki, Finland, August 2012.
[23] N. Spring, R. Mahajan, and D. Wetherall, Measuring ISP Topologies with RocketFuel, ACM SIGCOMM, Pittsburgh, USA, August 2002.
[24] N. Spring and D. Wetherall, A Protocol Independent Technique for Eliminating Redundant Network Traffic, ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.
[25] 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.
[26] 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.
[27] 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.
[28] Y. Zhu and M. Ammar, Algorithms for Assigning Substrate Network Resources to Virtual Network Components, IEEE Infocom 2006, Barcelona, Spain, April 2006.