Network Service Embedding for Cross-Service Communication

Angelos Pentelas and Panagiotis Papadimitriou

Department of Applied Informatics, University of Macedonia, Greece
{apentelas, papadimitriou}@uom.edu.gr

Abstract—Network Function Virtualization (NFV) facilitates the deployment and orchestration of network services (NSes) on virtualized infrastructures. NFV orchestration, in its prevailing form, deals with NSes independent to each other. This introduces significant limitations to cross-service interactions, i.e., a service that requires the consumption of (part of) another service. In an evolving service ecosystem that promotes cross-service interactions, orchestrating NSes with their cross-service communication (CSC) requirements into account is of crucial importance.

In this paper, we propose a CSC-aware NS embedding heuristic that optimizes NS placement, not only based on inner-component resource and communication demands, but also with respect to another deployed NS that will be consumed. To this end, we study potential types of CSC and introduce a new data structure, namely VNF embedding tree, which is used to generate the most appropriate embedding sequence. Assessing the efficiency of the proposed heuristic using simulations, we uncover significant gains in terms of service co-location without any perceptible embedding efficiency penalty, compared to a baseline heuristic that is oblivious of CSC.

I. INTRODUCTION

The advent of Network Function Virtualization (NFV) [1], [2], as a key ingredient for innovation in network processing, has brought virtual network functions (VNFs) to the spotlight of both researchers and practitioners. While NFV emerged to satisfy network processing requirements in the telco domain (i.e., softwarization of middleboxes, such as firewalls and NATs) [3], [4], its scope gradually expanded to cellular networks [5], [6] and other applications in various domains of social and economic activity (e.g., automotive, media, e-health, manufacturing), known as verticals [7].

This evolved service ecosystem creates opportunities for cross-service interactions, in the sense that one network service (NS), i.e., a sequence of VNFs [8], [9], can consume another NS, enhancing its functionality. For example, an augmented reality service offering overlaid metadata for current location could enrich its service delivery by providing personalized recommendations through a social network. Such services could be deployed using dedicated network slices within datacenters [10], [11]. This trend is gaining traction in edge computing for the support of latency-sensitive applications [7].

The orchestration of network services that need to consume others requires particular care. For example, NS embedding or scaling should be handled jointly with the consumed NS. More precisely, the embedding optimization of a NS does not solely pertain to resource/communication demands, but is also pertinent to the placement of the NS that will be consumed. However, the prevailing way of service orchestration deals with NSes independent to each other. As such, placement or scaling decisions are performed only with respect to the individual service/resource requirements. This approach can yield suboptimality in terms of CSC, i.e., the communicating components of the two services may be assigned to datacenter regions (e.g., different racks) with low-bandwidth connection or unnecessary long hop-count, increasing response time.

At first glance, CSC-aware NS embedding (NSE-CSC) could be presumably addressed by NSE methods that account for VNF sharing (e.g., [6], [12]); however, this is not the case. In particular, such studies consider the shared VNFs as an indispensible part of the service graph to be embedded and, if needed, new instances can be spawned within the physical infrastructure. In contrast, CSC implies that the shared VNF(s) comprise an exclusive property of a separate service, hence, the service which requires embedding has no means of replicating them on demand.

Along these lines, we stress on the need for NSE-CSC, with the aim of introducing a new embedding policy dimension, i.e., the co-location of the pair of communicating services. As such, NS embedding should not only take into consideration the various inner-component communication and resource demands, but should also strive to optimize the mapping of CSC links. This problem is further exacerbated by the potentially different types of CSC, depending on whether the entire or part of the service is consumed. Since existing NS embedding methods are, in principle, oblivious of CSC, we propose a new heuristic for the NSE-CSC problem. A key aspect of the proposed heuristic is the VNF embedding tree, i.e., a new data structure that aids the evaluation of VNF embedding steps, with the aim of generating the most suitable sequence for the embedding of a VNF-graph. Comparing the proposed heuristic with a baseline heuristic designed for generalized NS embeddings (thereby, not accounting for CSC), our proposed method yields a higher degree of communicating service co-location without any perceptible penalty in terms of embedding efficiency.

The remainder of the paper is organized as follows. In Section II, we elaborate on the notion of CSC and distinguish between three envisaged cases of CSC. Section III describes the problem at hand, with emphasis on the intricacies of CSC with respect to the NS embedding. Section IV introduces
our service models and presents the proposed heuristic, as well as the baseline. In Section V, we present our evaluation environment and compare the efficiency between the two heuristics. Finally, Section VI highlights our conclusions.

II. CROSS-SERVICE COMMUNICATION

Before delving into the NSE-CSC embedding problem, we lay out our perception of cross-service interactions. In this respect, we consider the following service classification: (i) consuming services, and (ii) providing services. Essentially, CSC provides the means for a consuming service to enhance its functionality by gaining access to service elements or functions offered by a providing service. Such services may be co-located in the same (edge) cloud infrastructure, opening up an opportunity for peering between the two services [10]. As such, the consuming service can be enabled to consume another (i.e., providing) service with very low latency, enhancing its functionality and potentially the experience offered to the user. For instance, Fig. 1 illustrates an augmented reality (AR) service co-located with a social media (SM) service. In particular, the AR service retrieves user preferences stored in the SM service; thus, it can offer an enhanced personalized AR experience to users. In this example, the AR is the consuming service, while the SM fulfills the role of the providing service. Henceforth, a pair of a consuming and a providing service is referred to as a CSC pair.

We distinguish between three types of CSC, illustrated in Fig. 2. More specifically, CSC type 1 represents a cross-service interaction, at which a certain VNF of the providing service is only consumed. In this case, a subset of the traffic traverses only the consuming service (i.e., VNF A → VNF B), whereas the rest of the traffic is redirected through the providing service (i.e., VNF A → VNF E → VNF B). For example, consider a mobile application as the consuming service, which offers basic and premium subscriptions. This subscription discrimination exposes basic users to advertisements, while, at the same time, providing an ad-free experience for premium users. In this scenario, VNF E of the providing service could be a virtualized cache that stores advertising content, which is offered to the consuming service. Consequently, traffic associated with basic subscriptions will traverse the path VNF A → VNF E → VNF B, instead of the path VNF A → VNF B, which will be utilized for premium users.

We further identify an additional CSC type, i.e., CSC type 2, at which the consuming service accesses a subset of the providing service, in the form of a sequence of VNFs, as depicted in Fig. 2. This CSC type pertains to services that require the consumption of a wider spectrum of functions offered by a providing service. For example, assume that VNF A is a firewall, whereas VNF E and VNF F correspond to a light and heavy intrusion detection system (IDS), respectively. In this case, flows that are suspicious for intrusion can be subjected to an additional two-level inspection by utilizing the respective VNFs of the providing service (VNF A → VNF E → VNF F → VNF B). Instead, all remaining traffic will traverse only the VNFs of the consuming service.

Last, we consider the scenario where the whole providing service needs to be consumed. For instance, assume that the providing service is a machine learning (ML) application, composed of VNFs that implement feature selection, model training, model testing, and classification, which practically constitutes an inseparable pipeline. This case is expressed by CSC type 3, at which the traffic traverses the entire chain of the providing service.

III. PROBLEM DESCRIPTION

The network service embedding problem (NSE) consists in the optimized mapping of virtual nodes and links comprising a VNF-graph, onto physical counterparts on a substrate network, e.g., a datacenter. Whereas the optimization objectives might slightly vary across research studies, the most common one is the minimization of the embedding footprint (e.g., [13], [14], [15], [16], [17]). This is practically translated into VNF consolidation, meaning that the VNF-graph should be mapped onto the minimum amount of servers, while generating the lowest possible amount of traffic. In this respect, inter-rack traffic becomes a highly undesirable implication, seeking its minimization given the oversubscription of datacenter network topologies. For instance, in Fig. 3, the providing service is mapped onto Servers 1 and 2. VNFs D and E, as well as VNFs F and G, communicate through virtual switching (e.g., OVS), while the remaining edge of the VNF-graph (i.e., VNF E → VNF F) is mapped onto a path that traverses the top-of-the-rack (ToR) switch, i.e., Switch 1. NSE, being a generalization of the NP-complete virtual network embedding (e.g., [18], [19]), is also classified as NP-complete, and its combinatorial nature requires greedy embedding methods in order to generate efficient solutions in polynomial time (unless P = NP).

The need to establish cross-service interactions introduces additional complexity and challenges into the NSE problem. The main challenge stems from the need to embed CSC links, which are marked with bold arrows in Fig. 3. These links establish the required connection between the services of a CSC pair. In case both services are instantiated concurrently, we could rely on existing techniques (e.g., [13], [14], [15], [17]) for their embedding, by merely bundling the respective
VNF-graphs. In reality, however, this is very unlikely; instead, the reasonable assumption is that the providing and consuming services will be deployed asynchronously. More precisely, we consider that the providing service is already in place (and consumed by its own dedicated clients), before the deployment of the consuming service. As such, the CSC VNFs of the providing service (e.g., VNF E for the CSC type 1) will already be fixed on the underlying network infrastructure. Thereby, the NSE optimization should handle these as additional constraints in the assignment of CSC links, and, in extension, in the mapping of the entire VNF-graph of the consuming service.

To explain this further, notice how the VNF-graph of the consuming service is expanded in the CSC type 1 scenario (top of Fig. 3). In particular, two CSC links are inserted, which connect VNF A with VNF E, and VNF E with VNF B, respectively. Given that VNF E is already placed (in Server 1, as shown at the bottom of the same figure), a single endpoint of each CSC link should be attached to Server 1. This introduces additional embedding limitations, compared to the generalized NSE problem, which are further exacerbated in CSC types 2 and 3. More specifically, the last two CSC types imply that four VNFs (two VNFs belonging to the providing and the other two VNFs being part of the consuming service) are involved in the CSC, whereas three VNFs are engaged in CSC type 1.

IV. SERVICE MODELS AND EMBEDDING HEURISTICS

A. Service Models

Network Service Model. A network service can be formulated as a directed graph $G = (V, E)$, which comprises vertices $i \in V$, representing VNFs, and edges $(i, j) \in E, i, j \in V$, expressing virtual links. Each vertex $i$ is associated with a CPU demand $d^i$, whereas each edge $(i, j)$ has bandwidth requirements, denoted by $d^{ij}$.

In the problem at hand, we denote the consuming service with $G_C = (V_C, E_C)$, and the providing service with $G_P = (V_P, E_P)$. According to the discussions in Section III, a CSC path comprises VNFs of both services (i.e., CSC VNFs), as well as links that do not exclusively pertain to any service (i.e., CSC links). We model $V_C = \{i, j\} (i, j \in V_C, i \neq j)$ as an ordered set that holds the CSC VNFs of $G_C$. Similarly, $V_P = \{k, l\} (k, l \in V_P)$ holds the respective VNFs of $G_P$. Note that, in the second case, it could be that $l = k$, corresponding to CSC type 1. We also model the set of CSC links by $E_{CSC} = \{(i, k), (l, j)\}$ $(i, j \in V_C, k, l \in V_P)$. The service model augments the modeling of the expanded consuming service (ECS), which is as follows:

**Expanded Consuming Service (ECS) Model.** ECS encompasses the vertices of the initial consuming service, plus the CSC VNFs of the providing service. Furthermore, ECS extends its edge set by adding the $E_{CSC}$ edges, i.e., ECS is modelled as a directed graph $G_C^* = (V_C^*, E_C^*)$, where $V_C^* = V_C \cup V_P$, and $E_C^* = E_C \cup E_{CSC}$. Each element of $G_C^*$ is associated with certain resource demands, similar to the case of the typical NS model.

B. NSE-CSC Heuristic

Towards the design of a heuristic algorithm, capable of coping with the complexity of NSE-CSC, we encounter a crucial issue, which stems from the ambiguity of VNF communication dependencies. That is, the CSC VNFs in $V_C$ will communicate with both their adjacent VNFs included in $V_C$, as well as with the assigned CSC VNFs of the providing service. From an algorithmic design perspective, if we prioritize these dependencies based only on communication requirements (e.g., bandwidth demands), we will neglect a key problem aspect, i.e., that we can capitalize on the mapping flexibility of unassigned VNFs. For instance, in Fig. 4a, VNF B should be placed in proximity to VNF A, VNF F, and VNF C, which are its adjacent nodes. While its most intense communication occurs with VNF A, prioritizing the placement of VNF B closer to VNF F (which is fixed), and then trying to map VNF A and VNF C close to VNF B, seems a viable approach to NSE-CSC.

Following this approach for all VNFs of a consuming service, we face a critical challenge of NSE-CSC, i.e., how to determine the VNF embedding sequence. This term refers
to the sequence, according to which, the embedding algorithm will seek to map the VNFS of the consuming service. Apparently, the VNF embedding sequence comprises a fundamental aspect of the overall embedding policy, since constructive heuristics, by definition, have no means of stepping back to modify the (partial) solution (although this feature obviously accelerates the solution computation). This means that, if the global embedding policy does not sufficiently address the problem at hand, the generated embeddings will be inefficient. Back in Fig. 4a, let us assume that the embedding commences at the root of the tree, increases the probability of co-locating the nodes of entire sub-trees rooted at X. The next step is to obtain VNF B’s adjacency VNFS (that belong to the consuming service). These are VNF A and VNF C and, thereby, these are added in the tree as VNF B’s children. Given that \( d^{AB} = 30 > 25 = d^{BC} \), VNF A will be VNF B’s left child, whereas VNF C will be its right child. No further branching occurs below VNF A, since its only adjacent node is its parent, i.e., VNF B. On the other hand, there is a link between VNF C and VNF D, therefore, the latter is a child of the former. Similarly, VNF E is placed below VNF D, and this completes the tree depicted in Fig. 4b. Note that the proposed tree structure can be applied on non-linear consuming service graphs, as well.

**Inferring the VNF embedding sequence.** The embedding sequence is derived through the following logic: the root of the tree will always be placed towards the already embedded node \( e \in V_P \) that it communicates with, while the rest of the nodes will be placed towards their parent.

The VNF embedding sequence is dictated by the visiting order of the tree’s nodes, when these are traversed with a depth-first strategy, prioritizing left sub-trees. This, in conjunction with the core logic described above, will result in the embedding sequence illustrated in Fig. 4c for the tree of Fig. 4b. In particular, the heuristic will initially seek to place VNF B as proximate to VNF F as possible. Subsequently, it will try to co-locate VNF A with VNF B, and then VNF C with VNF B. Likewise, VNF D will be sought to be mapped close to VNF C. Finally, VNF E’s mapping will depend on the placement of VNF D.

**VNF co-location policy.** The proposed embedding heuristic strives to co-locate VNF X towards an already placed VNF Y, meaning that it will first attempt to place X in the same server that Y resides. If this is not possible, the heuristic will attempt to embed X in a server belonging to the same rack; otherwise, it will place X in a server within the rack with the least outbound inter-rack traffic. The candidate servers of a rack are selected with a Worst Fit logic, if X is not a leaf node, since additional VNFS will pursue their co-location with X (i.e., X’s children). If X is leaf, a Best Fit approach is followed, since no additional VNFS will seek to communicate with X, thus, the algorithm should allocate CPU as efficiently as possible.

C. Baseline Heuristic

The baseline heuristic maps a NS into a datacenter as follows. Initially, it sorts the racks of the datacenter in descending order, according to their average available ToR-to-core switch link bandwidth. Commencing with the first ordered rack, it ranks its servers in descending order, according to their available CPU capacity, and strives to place VNFS.
TABLE I: Substrate Network Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of core switches</td>
<td>5</td>
</tr>
<tr>
<td>Number of racks</td>
<td>10</td>
</tr>
<tr>
<td>Number of servers / rack</td>
<td>20</td>
</tr>
<tr>
<td>Server CPU capacity</td>
<td>7.2 GHz</td>
</tr>
<tr>
<td>Intra-rack link capacity</td>
<td>1 Gbps</td>
</tr>
<tr>
<td>Inter-rack link capacity</td>
<td>10 Gbps</td>
</tr>
</tbody>
</table>

TABLE II: NS Request Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of VNFs / NS</td>
<td>U[3,8]</td>
</tr>
<tr>
<td>VNF CPU requirements</td>
<td>U[1,2,3,4,5] x 7.2 GHz</td>
</tr>
<tr>
<td>Link bandwidth requirements</td>
<td>U[10,100] Mpbs</td>
</tr>
</tbody>
</table>

TABLE III: Evaluation Environment Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSE requests / time interval</td>
<td>Poisson(20)</td>
</tr>
<tr>
<td>NS type</td>
<td>expiring</td>
</tr>
<tr>
<td>NS lifespan</td>
<td>U[3,10] time intervals</td>
</tr>
<tr>
<td>Number of time intervals</td>
<td>600</td>
</tr>
</tbody>
</table>

sequentially, starting from the first VNF of the chain. The VNF co-location policy is exercised in the same way with the NSE-CSC heuristic.

V. EVALUATION

In this section, we perform a comparison between the proposed NSE-CSC heuristic and the baseline variant, using simulations. Initially, we present our evaluation environment and metrics (Section V-A), and, subsequently, we proceed with the discussion of our evaluation results (Section V-B).

A. Evaluation Environment

We have developed a simulation environment for NSE evaluations in Python [20]. All evaluations are carried out on a two-layer fat-tree datacenter network topology. The datacenter consists of 200 servers, which are evenly arranged into ten racks. The corresponding ten ToR switches are interconnected through five core switches. The intra-rack and inter-rack links have capacity of 1 Gbps and 10 Gbps, respectively. The CPU capacity of each server is set to 7.2 GHz (see Table I).

In order to evaluate the NSE-CSC efficiency, we pre-embed ten providing services into the datacenter, with each one placed within a single rack. In this way, we do not preclude the existence of efficient solutions (which would be the case, if the VNFs of a providing service spanned multiple racks). In this respect, we generate consuming services that comprise three to eight, sequentially connected, VNFs. Each VNF requires 10, 20, 30, 40 or 50% of a server’s CPU in order to process its incoming traffic. We assume that bandwidth requirements vary for each virtual link in the range of 10 Mbps to 100 Mbps. Furthermore, each consuming service requires communication with a single providing service, which can be of either type 1, 2 or 3, as described in Section II. The aforementioned parameters, summarized in Table II, are obtained from uniformly random distributions.

Towards a more realistic simulation environment, we account for a set of $n$ discrete time intervals $T = \{1, 2, \ldots, n\}$. At each time interval, a number $N$ of consuming services request placement sequentially, where $N \sim \text{Poisson}(20)$. Each NS comes with a lifespan $k$. Therefore, if a NS is embedded at time $t \in T$, it will expire at time $t + k \in T$. During our simulations, $k$ lies within $[3,10]$ (randomly), and $n = 600$.

We compare the two heuristic variants based on the following criteria: (i) co-location efficiency of adjacent VNF pairs (i.e., how proximate two communicating VNFs are placed), (ii) embedding efficiency of CSC and non-CSC links, (iii) amount of inter- and intra-rack generated traffic, (iv) request acceptance rate (ratio of successfully embedded NSes over the total number of embedding requests), and (v) CPU utilization, which corresponds to the amount of the total CPU allocated to all embedded NSes.

B. Evaluation Results

Initially, we examine the proximity of each pair of adjacent VNFs (i.e., VNFs connected with a virtual edge), after their placement within the substrate network. According to Fig. 5, the NSE-CSC heuristic maps $\approx 30\%$ of such VNF pairs into the same server, whereas the respective percentage for the baseline method is slightly above $20\%$. Regarding the VNF pairs mapped into the same rack (but not in the same server), the NSE-CSC heuristic outperforms the baseline variant, since the corresponding percentages exhibit a difference greater than $10\%$. We further examine the mappings of communicating VNFs in different racks. It becomes apparent that the baseline method unfavourably maps nearly half of such VNFs into different racks, while only a quarter falls into the same category for the proposed heuristic. Apparently, this phenomenon is partially or fully attributed to the superior capability of the NSE-CSC algorithm to handle the optimized placement of CSC links, as opposed to the baseline method that merely optimizes the placement of the consuming service graph.

In order to gain more insights and identify whether the proposed algorithm skews in favour of the embedding optimization of the CSC links to the detriment of the embedding optimization of non-CSC links, we present Figs. 6 and 7. In particular, Fig. 6 illustrates the CDF of the hop count of non-CSC links, i.e., the edges solely pertaining to the consuming service graph. We note that the two-layer hierarchical topology used throughout our simulations implies three possible hop count values for an edge connecting two VNFs, i.e., 0 (same server), 2 (different servers in the same rack), and 4 (different racks). Fig. 6 indicates a similar behavior of the two methods, with respect to the placement efficiency of non-CSC links. This is very important, since the embedding efficiency of the consuming service part of the ECS graph is similar for both methods, as most of the time (i.e., $\approx 80\%$) the virtual links are confined within a single rack.

However, according to Fig. 7, the heuristic variants perform differently on the mapping of CSC links. More specifically, with the NSE-CSC method, $\approx 75\%$ of these links are mapped on paths comprising at most two physical links, meaning that the respective VNF pairs are placed either on the same server or on different servers within the same rack. The corresponding percentage resulting from the baseline algorithm is roughly $10\%$, i.e., 9 out of 10 pairs that comprise both
Fig. 5: Distribution of adjacent VNF pairs into same server, same rack, and different racks.

Fig. 6: CDF of the hop count for non-CSC adjacent VNFs.

Fig. 7: CDF of the hop count for CSC adjacent VNFs.

Fig. 8: Ratio of inter-rack traffic to total traffic (total = inter-rack + intra-rack traffic).

Fig. 9: Acceptance rate of the NSE-CSC and the baseline heuristic.

Fig. 10: CPU allocation for the NSE-CSC and the baseline heuristic.

consuming and providing VNFs are placed across different racks. This alone confirms the insights gained from Fig. 5 about the baseline method, regarding the high percentage of VNF pair distribution into different racks.

The previous results indicate that NSE-CSC exhibits a higher degree of intra-rack consolidation, compared to the baseline. This is also corroborated by Fig. 8, where the ratio of inter-rack to total traffic across all time intervals is depicted, for both methods. The main insight here is the considerable gap (i.e., \(\approx 10\%\)) in terms of generated inter-rack traffic (relative to the total traffic) between the NSE-CSC and the baseline. Such inter-rack bandwidth conservation by NSE-CSC allows for higher levels of datacenter network oversubscription and, in turn, cheaper switching hardware at the core level of the datacenter topology. Besides the cost reduction, confining traffic within a rack can lead to lower response time and more predictable performance.

Last, we investigate the efficiency of the two heuristics in terms of acceptance rate and CPU allocation. Figs. 9 and 10 indicate that both methods exhibit similar levels of efficiency with respect to these criteria, with only marginal differences that slightly favour the NSE-CSC. The small gain of the NSE-CSC potentially stems from the way it alters the ranking strategy of servers (Worst / Best Fit), since the baseline method does not incorporate any such feature.

According to our evaluation results and the aforementioned discussions, the proposed NSE-CSC heuristic manages to optimize the placement of ECs, without any penalty on the embedding efficiency of the consuming service part of the graph, or the resource allocation efficiency of the substrate network. Considering also the achievable optimization of the embedding of CSC links, we deem NSE-CSC as a viable and efficient solution to the NSE-CSC problem.

VI. CONCLUSIONS

In this paper, we tackled the challenging problem of NSE-CSC, i.e., the optimization of NS embedding, subject to CSC requirements. To this end, we designed a new heuristic that introduces a new policy dimension to generalized NS embeddings, i.e., the co-location of the pair of providing and consuming service. In order to tailor embedding optimization to both CSC and resource demands, we introduced the VNF embedding tree in order to derive the most suitable VNF embedding sequence for optimized NSE-CSC. Our evaluation results uncover significant gains for the proposed heuristic, in terms of service co-location. The proposed heuristic, most of the times, achieves the co-location of the CSC pair within a single rack (and sometimes within the same server), ensuring lower service response times, without any perceptible embedding efficiency penalty. Future work will be focused on the investigation of NSE-CSC using an experimental setup (e.g., OpenStack and OSM).

VII. ACKNOWLEDGMENTS

This work is supported by the MESON (Optimized Edge Slice Orchestration) project, co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code: T1EDK-02947).
REFERENCES