A Delay-Aware Approach for Distributed Embedding Towards Cross-Slice Communication

Ioannis Dimolitsas, Dimitrios Spatharakis, Dimitrios Dechouniotis, Symeon Papavassiliou
School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece
{jdimol, dspatharakis, ddechou}@netmode.ntua.gr, papavass@mail.ntua.gr

***Abstract*—The evolution of modern applications and the adoption of the 5G network slicing architecture require minimizing the communication delays. Virtual Network Functions (VNFs) offer low latency services to remote users. In the context of Edge Computing, these virtualized services are hosted in Edge Clouds, which, however, have limited resources. Embedding the Network Slice requests from users is a challenging problem, and demands distributed solutions in the presence of resource and delay constraints in the Edge Network. This paper introduces a distributed delay-aware service discovery mechanism, which performs cache-based forwarding, trying to efficiently discover VNFs in the Edge Clouds (ECs) and enable Cross-Slice Communication (CSC); that is the sharing of common VNFs between different Network Slices (NSes). Furthermore, the distributed NS embedding (NSE) problem is addressed by a modified k-Shortest Path (k-SP) approach to minimize the total round-trip network delay under the capacity constraints of the underlying infrastructures. The proposed method significantly decreases the execution time by 70% to provide the distributed NSE solution compared to the baseline k-SP algorithm.**

***Index Terms*—Network Function Virtualization, Network Slice Embedding, Service Discovery, Resource Management, Edge Computing**

I. Introduction

Edge computing is the emerging service delivery paradigm that addresses the resource orchestration challenges and guarantees the performance requirements of complex 5G verticals [1]. Similar to the cloud computing case, virtualized services are deployed over the computing infrastructure at the network edge to ensure low-latency communication with end-users. However, due to finite computing resources, the distributed placement of Network Slices that consist of multiple VNFs is required.

The current resource orchestration platforms facilitate the creation of Network Service Marketplaces [2], which allows users to be simultaneous both service providers and consumers. Similarly, 5G network slicing aspires to support smart manufacturing [3], where the orchestration of application is outsourced to micro-operators and relies on ECs. This complex environment requires the development of trust-worthy resource orchestration mechanisms that overcome any security constraints and enables zero-touch resource management for supporting multiple tenants.

These new business challenges dictate the transformation of the Network Function Virtualization technology. Initially, the strict VNF definition, which refers only to network-oriented functions such as firewall, routing, and load balancing, must be enriched with application-oriented functions. Secondly, the multi-tenancy of VNFs in such a way that the network slice isolation is preserved can be beneficial for both infrastructure providers and slice owners [4]. Towards this direction MESON platform proposes a centralized embedding [5], based on various functional and non-functional criteria, that places the entire network slice in a single EC aiming at maximizing the co-location between shared services between different slices. Contrary to centralized approaches, the distributed embedding solution places parts of the service chain among different ECs. Towards this direction, the DistNSE mechanism provides service chain partitioning across various infrastructure providers using an auction algorithm respecting the individual provider's policy [6].

This paper proposes a solution to the distributed embedding problem, which enables secure CSC between the network slices of different tenants. A Network Slice Embedding (NSE) request is assumed, which consists of multiple VNFs with different resource constraints that must be placed in the ECs. The major contributions of our work can be summarized as follows:

- By extending our previous work [7], we propose a distributed delay-aware service discovery mechanism that enables the discovery of CSC-enabled VNFs that are requested in the NSE request, minimizing the communication overhead within the Edge Network by utilizing a cache-based forwarding policy.
- Based on the results of the service discovery, an augmented network is constructed and a modified k-SP algorithm is proposed, to provide a solution to the distributed embedding problem. The proposed heuristic solution significantly reduces the time complexity of the baseline k-SP algorithm to minimize the round-trip network delay of the embedding solution.
- The proposed algorithm produces a similar delay for the NSE problem to the baseline algorithm for the same k values, however, outperforms it, in terms of execution time.

The remainder of the paper is structured as follows. Section II describes the system modeling and the formulation of the embedding problem. Section III provides the details of the distributed service discovery mechanism, while in Section IV an analytical description of the distributed NS embedding approach is presented. In Section V, we present the experimental

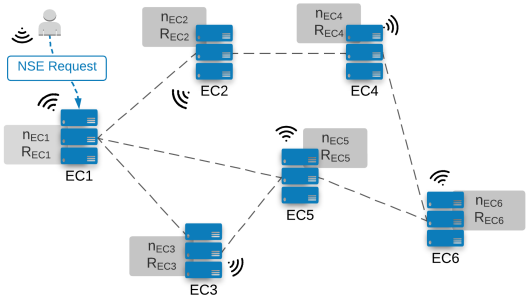

Fig. 1. Edge Network Overview

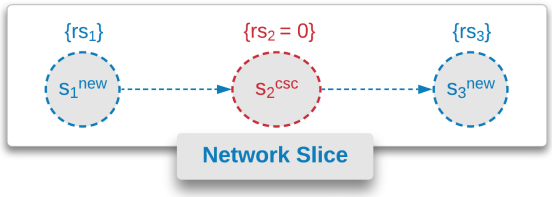

Fig. 2. Network Slice Embedding request with a CSC-enabled VNF in the chain.

evaluation of the proposed work compared with the baseline k-shortest path approach. Finally, Section VI concludes the paper and discusses our future plans.

II. System Modeling & Problem Description

This section describes the parameters of the Edge Network and the distributed Network Slice Embedding problem. Figure 1 illustrates the Edge Network topology, which consists of geographically distributed ECs. Slice owners submit their NSE requests in any EC in their vicinity. In this paper, towards enabling CSC, an NSE request dictates the placement of an NS, which consists of both new VNFs and CSC-enabled VNFs of different slice owners, as Fig. 2 shows. Each EC encompasses a mechanism that is responsible for providing an embedding solution that satisfies the resource requirements for the deployment of the new VNFs and the discovery of the CSC-enabled VNFs. The demand for high-quality service provisioning at the network edge necessitates minimizing the network delay. Considering the fact that an NS provides application functionalities to an end-user in a specific geographical region, we assume that the corresponding NSE request is submitted to a certain EC, termed as *Search Node* (SN). In this respect, the proposed Distributed Slice Embedding Mechanism (DSEM) focuses on providing a solution for the NS placement over different ECs, while minimizing the round-trip network delay. We define the round-trip delay as the sum of the following three terms; (a) the in-going delay from the Search Node to the first VNF of the NS, (b) the sum of the delays between the VNFs of the NS, and (c) the out-going traffic from the last VNF to the Search Node, through which the end-user application is provisioned. Moreover, depending on the available computing resources of each EC, parts of the NS could be co-located, enabling further reduction of the delay in the VNF chain. In this regard, we consider the Intra-EC delay between VNFs to be negligible.

A. Edge Network Model

We illustrate the Edge Network as a Graph $G = (V, E)$. The set V corresponds to the nodes of G and $v_i \in V$ represents the EC_i of the Edge Network. The network links between two ECs of the Edge Network constitute the set E and are denoted as $e_{ij} = (EC_i, EC_j)$. Besides that, we denote by d_{ij} the delay of the network link between EC_i and EC_j. In addition, for each $v_i \in V$ the available computing resources are denoted as R_v representing CPU cores and the provided CSC-enabled services n_v are attributes of an EC $v \in V$. An example of the an NSE request, to the SN EC_1, in the Edge Network is illustrated in Fig. 1, while the corresponding graph representation is shown in Fig. 3.

B. Network Slice Embedding Request

A slice tenant submits an NSE request in a specific node $v \in V$. The request consists of 1) the NS template, which is represented as a directed service graph $G_s = (V_s, E_s)$, where $V_s = \{s_i^a\},\ i = 1, 2, \ldots, M$ represents the VNFs of the NS and the set E_s includes the links between the VNFs, and 2) the set $RS = \{rs_i\},\ i = 1, 2, \ldots, M$ that corresponds to the resources requested for each VNF and it is expressed in CPU cores. We denote by $s_i^a \in V_s$ the i^{th} VNF of the NS, where when $a = csc$ corresponds to a CSC-enabled VNF requested by the slice tenant and $a = new$ corresponds to new deployed VNFs. In case of an already deployed CSC-enabled VNF, i.e., s_i^{csc}, the corresponding resource demand is $rs_i = 0$. Furthermore, the slice tenant can define an upper bound about the round-trip network delay as a strict requirement of the submitted NSE request. In essence, this requirement determines the acceptable network delay level, for the corresponding application to be operational, based on the slice tenant's preferences and it is denoted by D_r.

C. Problem Statement

The above discussed CSC-enabled Network Slice Embedding problem is formulated according to the Edge Network and NSE request specifications, as these determined previously. Since the NSE request consists of both new deployed VNFs and shared VNFs (i.e., s_i^{csc}), which are already hosted in some ECs, the problem can be broken down into two sub-problems; at first, the discovery of the deployed CSC-enabled VNFs and then, the placement and deployment of the rest, new VNFs of the NS, based on their requested resources, aiming to the minimization of the total round-trip network delay D. Therefore, the problem is summarized as follows. Given the Edge Physical Network G with V nodes and E links, the NSE request represented as a directed service graph G_s, with a length of M VNFs $s \in V_s$ with the corresponding resource requirements RS and D_r the maximum allowable round-trip

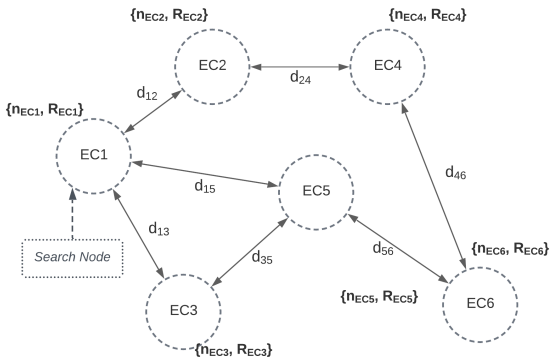

Fig. 3. Edge Network's Graph Representation

delay, the objective of the proposed approach is to provide a NSE solution U that minimizes the round-trip network delay. We, also, denote the Search Node SN as $u_{sn} \in V$, where the NSE request is submitted. A solution U^j is defined as:

$$U^j = \{u_0^j, u_1^j, u_2^j, \ldots, u_M^j, u_{M+1}^j\},$$

where $u_i^j \in V$, is the EC that the VNF s_i is embedded in the solution U^j. So, we say that u_i^j is the embedding node of s_i. Furthermore, it stands that $u_0^j = u_{M+1}^j = u_{sn}$, as the network traffic has to return back to the end-user via the Search Node. Let $\mu_{i,i+1}^j$ is the shortest path cost in G between two embedding nodes u_i^j, u_{i+1}^j of U^j that corresponds to the network delay. Furthermore, the parameter $\epsilon_{i,v}$ determines whether an EC v is the embedding node for VNF s_i, where,

$$\epsilon_{i,v} = \begin{cases} 1, & \text{if } u_i = v \\ 0, & \text{otherwise} \end{cases}, i = \{1, \ldots, M\}$$

The round-trip network delay of a solution U^j is defined as follows:

$$D_{U^j} = \sum_{i=0}^{M} \mu_{i,i+1}^j. \tag{1}$$

Assuming that $\mathcal{U} = \{U^j\}$ is the set of the embedding solutions, our objective is to minimize the total round-trip network delay D_U of the NSE:

$$\min_{U^*} D_U \Rightarrow \min_{U^*} \{\sum_{i=0}^{M} \mu_{i,i+1}\} \tag{2a}$$

Subject to:

$$\sum_{i=1}^{M} \epsilon_{i,v} r s_i \leq R_v, \ \forall v \in V \tag{2b}$$

$$u_0 = u_{M+1} = u_{sn} \tag{2c}$$

$$U^* \in \mathcal{U} \tag{2d}$$

The constraint (2b) ensures the resource availability in every EC v that is an embedding node for the respective VNFs. The equation (2c) makes sure that the round-trip network traffic will return to the Search Node after the VNFs operations within the Network Slice, while (2d) implies that the optimal solution belongs to the solution set $\mathcal{U}$.

III. CSC-Enabled Service Discovery

An efficient discovery of CSC-enabled VNFs in the Edge Network is of major importance for optimizing distributed NSE solutions with CSC capabilities. The proposed distributed CSC-enabled service discovery (CSC-SD) considers several parameters. These parameters refer both to the ECs and the NSE requests. Specifically, the discovery is based on the *CSC VNFs Cache*, which is an element of each EC and determines the request forwarding policy at every hop.

A. CSC-Enabled VNFs Cache

Each EC maintains a cache, which contains records about the most recent hosted CSC-enabled VNFs for all neighbouring ECs. For instance, the cache of the EC_1 of Fig. 3 contains the corresponding records about the EC_2, EC_3 and EC_5. Since the NSE request can be submitted in any EC, the cache content alongside with the D_r round-trip delay requirement determines the forwarding policy of the request. More specific, an EC forwards the request to a neighbor EC, whether its cache entries contain at least on of the demanded CSC-enabled VNFs, otherwise it broadcasts the request to all adjacent ECs.

Algorithm 1 CSC-enabled Service Discovery

```
procedure DISCOVERY(G, T, source, csc, D_current)
    if D_current < 0 then
        return T
    else
        forwardToNodes = Cache-based CSC Forwarding
        for node in forwardToNodes do
            if edge e(source, node) not in T then
                if d_e < D then
                    ADD e(source, node) in T
                    D' = D_current − d_e
                    T=Discovery(G, T, node, csc, D')
                end if
            end if
        end for
        return T
    end if
end procedure
```

B. Delay Aware Distributed Service Discovery

Starting from the Search Node, the discovery process following the described policy, and a query message, which contains all the requested CSC-enabled VNFs $\{s_i^{csc}\}$ and the slice tenant's round-trip delay requirement D_r, is forwarded. Contrary to other discovery techniques that broadcast the query messages with a specific Time-To-Live (TTL) value [7], in the proposed approach, the search depth in the graph G is determined by the D_r value, while the message forwarding is performed based on the cache entries of each EC.

Algorithm 1 performs the discovery of the s_i^{csc} VNFs of the NSE. The algorithm's output is a search graph T that contains paths which lead to ECs that occurred as candidate embedding nodes for one or more of the s_i^{csc}. At each step, the algorithm examines if the current delay threshold of the search tree $D_{current}$ is greater than zero (line 2). In this case, it forwards the query message based on the cache entries (line 5) and adds the corresponding edge in the search tree, while updates the current delay threshold value D' (lines 6-14). The implementation of the specific discovery approach aims to mitigate the network communication overhead, compared to centralized broadcast approaches. Furthermore, the provided search graph T, which is a sub-graph of G, down-scales the search space for providing NSE solutions.

IV. Distributed Network Slice Embedding

In this section, the methodology for solving the above described minimization problem is presented. Two different shortest path-based methods are implemented, trying to balance between a good approximation for the embedding solution, and the complexity demands of the problem. Besides that, individual features of the approaches are combined, in order to design an efficient methodology in terms of solution convergence and computational requirements. Also, we attempt to take advantage of the search graph T provided by the CSC-Enabled service discovery, to obtain a good approximation of the optimal solution.

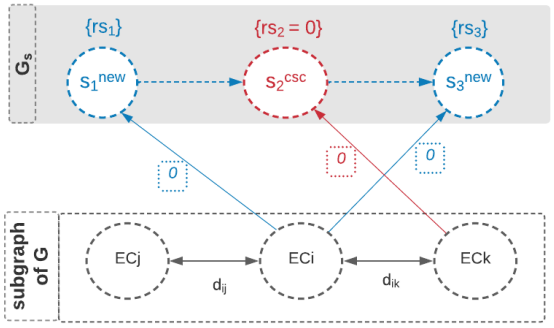

Fig. 4. Augmentation of G For Shortest Path-Based Approach

A. Graph Augmentation

For the purpose of providing an embedding solution using shortest paths algorithms, we perform an augmentation of the network model G as follows. Initially, we introduce new nodes in the model for each VNF of the NSE request. The augmented graph is declared as $G^{'}$ and it contains M new nodes, which refer to the $\{s_i^a\}$, $i = 1, \ldots, M$ VNFs. According to the required VNF resources rs_i, the available resources R_v of each EC $v \in V$ and the service discovery outcome, a virtual link with zero delay is added between an EC node $v = EC_i$ and a VNF node $u = s_j^a$. In precise, an edge between a VNF s_i^a and an EC node v is added in $G^{'}$ if s_i^a is CSC-enabled VNF (s_i^{csc}) and is hosted to node v based on the service discovery, or s_i^a is a VNF to be deployed (s_i^{new}) and $rs_i \leq R_v$. It is worth mentioning, that for the new deployed VNFs s_i^{new}, the augmentation is performed individually for each VNF and each EC is considered as a candidate for the placement of the corresponding VNF. So, the VNF resource demands rs_i for each s_i^{new} are compared with R_v, regardless previous addition of virtual edges. The capability of co-location for two or more VNFs is dissected during the NSE process, where, after each VNF embedding, the respective available resources of the EC are updated.

Algorithm 2 Dijkstra-based Approach

1: **Inputs:** $G^{'}, u_{sn}, V_s,$
2: **Outputs:** U : embedding solution, D_U : round-trip delay
3: Initialize $D_U \leftarrow 0$, $u_0 = u_{sn}$
4: **for** $\forall$ s_i^a in V_s **do**
5: Identify host u_i from DijkstraPath(u_{i-1}, s_i^a)
6: Add u_i in U
7: Update available resources R_{u_i} after embedding
8: Update virtual links from u_i in $G^{'}$
9: **end for**
10: $u_{M+1} = u_0$
11: Calculate D_U from Eq. (1)
12: **return** U, D_U

B. Dijkstra Shortest Paths-Based Approach

At first, we implement a greedy approach based on Dijkstra's shortest path algorithm (Greedy Dijkstra Embedding Approach - GDEA). Considering the list of nodes of $G^{'}$, which represent the NS: $V_s = s_1^a, \ldots, s_M^a$, at each step of the algorithm, we calculate a Dijkstra path, with source an EC node v of $G^{'}$ to the next VNF node s_i^a, that has to be embedded. In the beginning, the first path that is calculated is the $u_{sn} = u_0$ to the s_1^a, and the embedding node u_1 is identified. Subsequently, the paths that are calculated at each step are (u_i, s_{i+1}^a). As this Dijkstra path lead to a VNF node via a virtual link, the embedding node u_i, which the s_i^a is hosted, is the second to last node of the Dijkstra path. Afterwards, the available resources of the EC have to be updated and remove the virtual links to VNFs with resource demand higher than the updated EC's resources. Algorithm 2 shows the discussed implementation of the GDEA. The time complexity of Dijkstra algorithm is $O((|V|+|E|)log|V|)$. So, considering that GDEA uses the Dijkstra algorithm for M times, the time complexity of this approach is $O(M(|V| + |E|)log|V|))$. Although the GDEA provides an embedding solution in limited time, it does not always provide a good approximation to the optimal solution.

C. k-Shortest Paths Embedding Heuristic

An exhaustive search, for finding the optimal solution, is characterized from an extremely high computational complexity. In order to accomplish a better approximation to the optimal solution compared to GDEA, a k-SP-based Heuristic,

termed as k-Shortest Path Embedding Heuristic (k-SPE-H), is introduced. The proposed heuristic is based on the Yen's algorithm [8] that finds the k-shortest loop-less paths in a graph. Various studies regarding virtual network embedding problems utilize this algorithm, such as in [9], in order to deal with the exponential solution space of the problem, while achieving a good approximation. However, the optimality of the solution strongly depends to the k value, which determines the number of the examined paths for embedding. So, as k increases, the paths for embedding an NS with M VNFs that will be examined are k^M. To achieve better approximation with a higher value of k and eliminate the number of examined paths, we propose a heuristic that in each recursion determines a delay upper bound of each examined solution U^j, which initially is provided from the above described GDEA solution. Algorithm 3 outlines the following process. Precisely, our heuristic calculates the shortest path of the last embedding node u_i^j from the source u_{sn}, and if the round trip delay is more than the determined upper bound L_D, the construction of this solution is terminated (lines 8-10). Starting from the Search Node u_{sn}, the k-shortest paths to the first VNF of the NS, $s_1^a \in V_s$, are calculated. For each path, we find the host EC for s_1^a, update the EC's capacity (lines 20-21). A solution U^j is constructed and the procedure is then called in a recursive way to find the path for the next VNF of V_s (lines 23-27). When there are no more VNFs to be embedded, the solution U^j is stored in the set of solutions $\mathcal{U}$, and the upper bound delay L_D is updated accordingly (lines 12-16). The time complexity of Yen's k-shortest paths algorithm is $O(k|V|(|V|+|E|)log|V|)$. Thus, as k-SPE-H uses the Yen's algorithm to produce k-shortest paths M times, the worst case complexity of this approach is $O(Mk|V|(|V|+|E|)log|V|)$.

Algorithm 3 k-SPE-H Approach

1: **Inputs:** $G^{'}$, u_{sn}, V_s, k
2: **Outputs:** U^*: NSE solution, D_U: round-trip delay
3: Initialize: $\mathcal{U}, U^j \leftarrow \emptyset$
4: $L_D \leftarrow$ algorithm's 2 solution round-trip delay
5: $\mathcal{U}$ = k-SPE($G^{'}, V_s, u_{sn}, L_D, \mathcal{U}, U^j$)
6: $D_{U^*} \leftarrow$ (2a), where $U^* \in \mathcal{U}$
7: **procedure** K-SPE($G^{'}$, V_s, u_{i-1}, L_D, $\mathcal{U}$, U^j)
8: **if** $D_{U^j} \leq L_D$ **then**
9: Stop constructing this solution
10: **return** $\mathcal{U}$
11: **end if**
12: **if** V_s is empty **then**
13: Add u_{M+1}^j in U^j
14: Append U^j in $\mathcal{U}$
15: Update L_D delay upper bound
16: **return** $\mathcal{U}, L_D$
17: **else**
18: Identify next VNF's s_i^a host u_i^j
19: Add u_i^j in U^j
20: $V_s^{'} \leftarrow$ extract s_i^a from V_s
21: Update resources $R_{u_i^j}$ in $G^{'}$
22: Update virtual links from u_i in $G^{'}$
23: $\mathbf{K} \leftarrow$ find k-shortest paths from u_i^j to s_{i+1}^a
24: **for** $\forall\ \kappa_i \in \mathbf{K}$ **do**
25: Identify embedding host u_{i+1}^j for s_{i+1}^a from κ_i
26: ▷ Continue for the next VNF in $V_s^{'}$:
27: $\mathcal{U}, L_D = kSPE(G^{'}, V_s^{'}, source, L_D, \mathcal{U}, U^j)$
28: **end for**
29: **return** $\mathcal{U}$
30: **end if**
31: **end procedure**

The second experiment aims to highlight the effectiveness of the proposed Distributed CSC-Enabled Service Discovery (CSC-SD). To this end, we compare the k-SPE-H solutions average round-trip delay, when the CSC-Enabled VNFs are discovered (1) through the proposed CSC-SD, and (2) using a centralized broadcast Service Discovery (B-SD), where the query message is forwarded to each of the neighbors nodes of every sender EC and it is used. Figure 6 demonstrates the

V. Evaluation

The evaluation of the proposed distributed NSE approach is discussed in this section. For the experiments, we rely on network topologies from [10], and we perform simulations on an edge network with 48 ECs (nodes), using the NetworkX Python package [11]. Each EC hosted between 5 to 10 CSC-enabled VNFs, from a pool of 20 in total. The caches for every EC have a standard initial configuration for all the NSE requests, where its size is equal to 5, which is 25% of the available CSC-enabled VNFs. The network links delays, d_{ij} vary form 5 to 10 ms following a uniform distribution, while the EC capacity, in available CPU Cores, R_v ranges uniformly from 4 to 12. For varying NSE request sizes (3 to 6), we generated 100 different requests. An NSE request consist of 1 or 2 CSC-enabled VNFs, while the rest VNFs regard to new deployments with demands rs_i 2 to 6 CPU cores uniformly distributed. Every NSE request is submitted to a specific EC in the edge network. The requested round-trip network delay D_r ranges uniformly from 60 to 100 ms.

Under this configuration of the Edge network, two different experiments are selected to showcase the performance of the proposed technique. In the first experiment we compare the efficiency of the proposed k-SPE-H over the baseline k-SPE solution as it is presented in [9]. With the values of k equal to 5, 6 and 7, both algorithms strive to provide an embedding solution for each request. Both approaches achieve equal results in the round-trip delay. However, as it shown in Figure 5, k-SPE-H provides the embedding solution much faster than the baseline k-SPE for all the k values, while avoids the exponential growth of the execution time as NSE length M increases. Specifically, the execution time reduction reaches the 70% in average for higher NSE lengths. This is achieved as the k-SPE-H calculates the distance of each new embedding node u_i^j from the u_{sn} during the construction of an embedding solution U^j and stops the expanding of U^j when the upper delay limit L_D is exceeded.

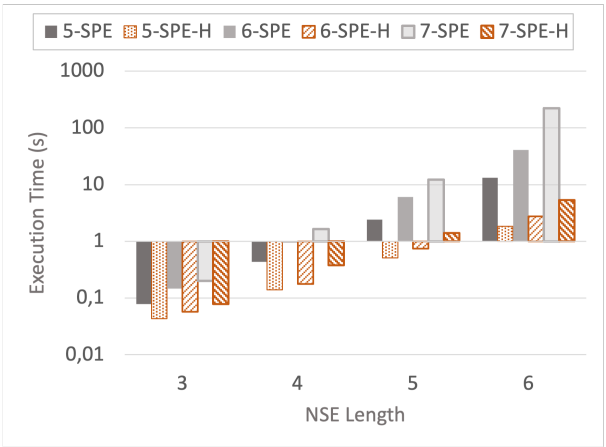

Fig. 5. Execution Time Comparison for several k values.

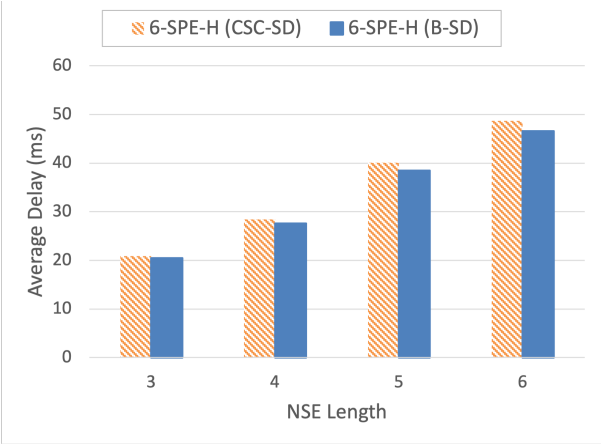

Fig. 6. Average Round-trip delay of 6-SPE-H using CSC-SD vs B-SD.

average round-trip network delay achieved in both cases. As it shown, in both cases the proposed heuristic provides similar solution in terms of round-trip network delay. For more than 90% of the requests, the algorithm achieves equal network delay using both service discovery techniques, while, in average, when B-SD is used leads to approximately 3% delay reduction in average, in comparison when CSC-SD is used. However, the CSC-SD utilizes only the 20-22 % of the physical links of the Edge network, minimizing the communication overhead that the discovery process adds, while providing the k-SPE-H with a reduced search space, to operate faster.

VI. Conclusions

This paper presents a distributed network slice embedding approach that consists of a distributed service discovery method and a heuristic k-shortest paths based approach and aims at minimizing the round-trip network delay of the embedded network slice. Furthermore, this approach enables cross-slice communication between different network slices that increase the resource utilization and further reduce the network traffic. The numerical results present the computational time benefits obtained by the combination of the fast greedy GDEA and the search tree of the CSC-Enabled Service Discovery with the k-shortest paths algorithm. That allows higher values of the k parameter to be determined, in order to obtain better approximations to the optimal solution. Regarding our future directions, we aim to extend the proposed method to deal with the VNF placement problem with resource allocation constraints within the EC infrastructure, combined with the distributed NSE. Furthermore, we will investigate a more efficient solution about distributed NSE problem, in terms of minimization of the round-trip network delay and the computational complexity.

Acknowledgment

This work was supported by the CHIST-ERA grant CHIST-ERA-18-SDCDN-003 (DRUID-NET), and is co-financed by Greece and European Union under the Operational Programme "Competitiveness, Entrepreneurship and Innovation" (EPAnEK) through the Greek General Secretariat for Research and Innovation (GSRI), grant number T11EPA4-00022.

References

[1] D. Dechouniotis, N. Athanasopoulos, A. Leivadeas, N. Mitton, R. Jungers, and S. Papavassiliou, "Edge computing resource allocation for dynamic networks: The DRUID-NET vision and perspective," *Sensors*, vol. 20, no. 8, p. 2191, 2020.

[2] L. Bondan, M. F. Franco, L. Marcuzzo, G. Venancio, R. L. Santos, R. J. Pfitscher, E. J. Scheid, B. Stiller, F. De Turck, E. P. Duarte, *et al.*, "FENDE: marketplace-based distribution, execution, and life cycle management of VNFs," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 13–19, 2019.

[3] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5g network slices to support industrial internet and to shape next-generation smart factories," *IEEE Network*, vol. 33, no. 4, pp. 146–154, 2019.

[4] G. Papathanail, A. Pentelas, I. Fotoglou, P. Papadimitriou, K. V. Katsaros, V. Theodorou, S. Soursos, D. Spatharakis, I. Dimolitsas, M. Avgeris, *et al.*, "MESON: Optimized cross-slice communication for edge computing," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 23–28, 2020.

[5] I. Dimolitsas, D. Dechouniotis, V. Theodorou, P. Papadimitriou, and S. Papavassiliou, "A multi-criteria decision making method for network slice edge infrastructure selection," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–7, IEEE, 2020.

[6] A. Abujoda and P. Papadimitriou, "Distnse: Distributed network service embedding across multiple providers," in *2016 8th international conference on communication systems and networks (COMSNETS)*, pp. 1–8, IEEE, 2016.

[7] I. Dimolitsas, D. Dechouniotis, S. Papavassiliou, P. Papadimitriou, and V. Theodorou, "Edge cloud selection: The essential step for network service marketplaces," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 28–33, 2021.

[8] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[9] N. Torkzaban and J. S. Baras, "Trust-aware service function chain embedding: A path-based approach," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 31–36, IEEE, 2020.

[10] The University of Adelaide, "The Internet Topology Zoo." http://www.topology-zoo.org/index.html.

[11] "NetworkX, Network Analysis in Python." https://networkx.org, Last Accessed on 2022-07-10.