SPEED - SFC Placement in Edge-Cloud Continuum: a Distributed Approach

Anselmo Luiz Éden Battisti[1], Flavia C. Delicato[2], Débora C. Muchaluat-Saade[2]

[1]MídiaCom Lab, Institute of Computing, Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/n, São Domingos, Niterói, RJ, 24210-590, Brazil

{anselmo, debora}@midiacom.uff.br, fdelicato@ic.uff.br

Abstract. *In the last decades, there has been a trend to virtualize computing and networking resources. This approach was initially adopted in the core functions of the network, thus creating the Network Function Virtualization paradigm. The process that defines the computational nodes and links to execute a set of Virtual Network Functions (VNF) is called placement. However, executing only VNFs individually is often not enough to meet users requirements. Therefore, the Service Function Chain (SFC) concept was created. With the advancement of the Edge-Cloud continuum infrastructure, SFCs started to be executed on multiple nodes, sometimes managed by independent service providers. In this multi-domain scenario, a distributed placement mechanism to allocate SFCs without complete knowledge of the infrastructure is required. In this work, we propose a new approach named SPEED to solve the SFC Placement Problem over a multi-domain environment in a distributed manner. The solution encompasses algorithms and a new mapping model between the SFCPP and the Game Theory approach. The results show that SPEED is feasible and, compared to other approaches, has a gain of 30% in the number of SFC requests placed.*

1. Introduction

In recent decades, there has been a trend to virtualize computing and networking resources over physical infrastructure [Mao et al. 2017]. This technology enabled the development of multiple computing environments such as cloud, edge, Internet of Things (IoT) and 5G [Pan and McElhannon 2018]. This virtualization paradigm was first applied to core network functions such as Network Address Translation (NAT), Firewall, and Deep Packet Inspector, thus creating the Network Function Virtualization (NFV) paradigm [Yi et al. 2018]. NFV decouples network functions from the underlying dedicated hardware and executes them through software, which is called Virtual Network Function (VNF) [Mijumbi et al. 2016].

Often, a single VNF is not capable of providing all the features to meet the user's service demands. Typically, users request complex services composed of multiple VNFs [Mostafavi and Hakami 2021], creating a new concept called Service Function Chain (SFC) [Kaur et al. 2020]. An SFC is a chain of sorted VNFs, associated with a Service Level Agreement (SLA) [Bhamare et al. 2016]. An SLA is a formal agreement between a service provider and a customer, defined by rules that outline the specific levels of service that the provider is required to provide [Kapassa et al. 2018]. These rules are derived from a variety of sources, including Key Performance Indicators (KPIs), low-level infrastructure demands, and high-level user requirements. For example, the SLA may specify the maximum acceptable SFC delay for response times.

Therefore, with the demand to place the multiple VNFs composing an SFC, the SFC Placement Problem (SFCPP) has emerged [Wang et al. 2020]. The SFCPP can be described as *giving a requested SFC, a set of conditions, restrictions, and data about the infrastructure that provides compute and network resources; the task involves identifying the compute nodes to execute each VNF and the network links that provide connectivity through the SFC* [Santos et al. 2022]. The contributions of this work are: i) A novel approach to solving the SFC Placement Problem in a multi-domain environment through a distributed method using Singleton Congestion Game; ii) A new strategy for dynamically segmenting the VNFs of the SFC, enabling execution across different domains; iii) A series of experiments using real network topologies to validate the performance of the proposed approach. The results demonstrate that SPEED improves the SFC allocation rate without increasing monetary costs.

2. Related Work

This section explores different approaches to solving the SFCPP. While centralized solutions are common, they face limitations regarding scalability, communication overhead, and privacy in multi-provider environments. Consequently, distributed research, depicted in Table 1, has surged to address these gaps.

Chen et al. [Chen et al. 2022] propose a decentralized SFC placement model where an ingress domain orchestrator coordinates resource allocation across independent domains using an aggregated network graph. The process involves a distributed ”SFC Partition” step, where the ingress orchestrator identifies the best inter-domain paths and delegates specific chain segments to local domain orchestrators for parallel execution. If a selected domain cannot fulfill its local placement, the system provides resilience by iteratively exploring the next best k-shortest paths in the multi-domain topology.

Another approach was proposed by Liu et al. [Liu et al. 2020]. They present a distributed SFC placement based on a distributed auction in a multi-domain environment. Each domain is independent and shares with the other domains only the peering nodes and which VNFs it can execute. Also, each domain can execute its own orchestrator configured to meet the service provider's interests.

Gang et al. [Sun et al. 2018] present a novel approach to solving the distributed SFC Placement in a multi-domain environment. In the adopted architecture, there is one main orchestrator that will coordinate the placement of all the SFC Requests. The resources of each domain are independent, and the domain only shares the peering nodes and the VNFs it can execute.

Table 1. Distributed SFC Placement approaches

Study	Technique	Environment	Segmentation	Orchestrator
[Chen et al. 2022]	Graph Theory	Edge / Cloud	Static	Distributed
[Liu et al. 2020]	Auction	Edge / Cloud	Static	Distributed
[Sun et al. 2018]	Mesh Aggregation	Edge / Cloud	Static	Centralized
[Avasalcai et al. 2019]	Auction	Edge	Static	Centralized
SPEED	**Game Theory**	**Edge / Cloud**	**Dynamic**	**Distributed**

3. System Model

In this section, we present the system model considered in our work. The network is modeled as an undirected graph $G = (\Omega, L)$. Ω denotes all the zones in the environment. A zone is defined as a logical unit that participates in the SFC placement process by managing a subset of computational and networking resources. A domain may contain one or more zones, depending on the administrative and topological structure of the environment. Zones are responsible for handling local metadata, evaluating placement possibilities, and interacting with other zones to collaboratively deploy service function chains in a distributed manner. L denotes the physical links between the domains.

The $Z = \{z_1, z_2, ..., z_n\}$ denotes all VNF types. Each $z \in Z$ has the attributes cpu_z and mem_z that define the CPU and memory required in the Compute Zone. The $\Re = \{r_1, r_2, ..., r_n\}$ denotes all SFC Requests. Each SFC Request $r \in \Re$ is an n-tuple defined as $r = (src, dst, V, VL, max\_delay, \Phi, \tau)$, where src and dst are the ingress and egress nodes respectively. $V = \{v_1, v_2, ..., v_k\}$ is the sorted list of VNFs, $v_i \in Z$, which make up the requested SFC. $VL = \{vl_1, vl_2, ..., vl_n\}$ is the sorted list of virtual links, $vl \in VL$ that connect the VNFs and are mapped to physical links. The virtual link vl_1 associates the user access node with the first VNF, and the virtual link vl_n associates the final VNF with the egress node. Each vl_i has the attribute bw that defines the required bandwidth of the virtual link. The max_delay represents the maximum tolerable network delay between src and dst. And finally, Φ, which prevents or imposes the deployment of a specific VNF z_i in a zone ω_j. The τ defines the time when the SFC Request arrives.

The set $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$ are the network zones. Each zone $\omega_i \in \Omega$ has the attributes: $childrens_\omega = \{\omega_1, \omega_2, ..., \omega_n\}$ which comprises subordinated zones, $\Lambda_\omega = \{z_1, z_2, ..., z_n\}$, $z_i \in Z$ which represents VNFs that can be executed by at least one subordinated zone, $parent_\omega$ which defines its parent zone, if $parent == NULL$ it means that the zone is the highest element in the hierarchy, $type_\omega$ identifies whether it is an aggregated or a Compute Zone. β_ω^z represents the cost of executing a VNF z in zone ω. The attribute $childrens_\omega$ of a Compute Zone is empty, and its resources are reserved to execute VNFs. The set $L = \{l_1, l_2, ..., l_n\}$ are the links between the zones. For every link $l_i \in L$, between two zones ω_1 and ω_2, we use bw_l, (bandwidth capacity) and $delay_l$ (delay). The parameter $\Psi_{l_j}^{vl_i}$ are the cost of executing the virtual link vl_i in the link l_j.

The SFC execution consumes resources, for which the service provider charges a fee. Therefore, our problem formulation requires the decision variable x_{ij}, which defines whether VNF z_i will be executed in zone ω_j, and the decision variable y_{ij}, which defines whether virtual link v_i will be executed in link l_j. Equation (1) defines how the cost of computational resources is calculated. Equation (2) defines how the cost of network resources is calculated. Thus, our objective is to minimize $CompCost + NetCost$. The problem is subject to: i)VNF can only be executed in Compute Zones, ii) VNF can only be executed in zones where the VNF Type is available.

$$CompCost = \sum_{z_i \in Z} \sum_{\omega_j \in \Omega} x_{ij} * \beta_{\omega_j}^{z_i} \tag{1}$$

$$NetCost = \sum_{v_i \in V} \sum_{l_j \in L} y_{ij} * \Psi_{l_j}^{v_i} \tag{2}$$

4. Proposed Solution

This section presents the SPEED approach. Our proposal creates a placement plan for an SFC request, based on the resources available in the environment and the SFC requirements. The placement plan maps each VNF to a compute node and the connections between the VNFs to virtual links. SPEED component is executed in a multi-domain environment. A domain is a set of computational and network resources owned by a service provider. Service providers compete against each other to lease their resources to execute the requested SFCs. The resources in the domains are hierarchically organized into zones. Zones are sets of compute nodes and network resources.

Our proposal differs from other approaches in terms of how the VNFs are assigned to each Compute Zone. In SPEED, the zone that initiates the placement process does not define the allocation of VNFs across Compute Zones. Rather, the Compute Zone that will execute each VNF is determined in a recursive manner. Once inside each zone, there is a component that collects data about which VNF Types can be executed, the monetary cost to execute each VNF Type, and the minimum delay to the border gateway of that zone. These metadata flow from the lower zones to the higher zones in the topology. Each level of the topology processes the metadata of the zones below it. We call this process metadata consolidation, and then the processed metadata is sent to the parent zone.

The workflow of our proposed approach begins with the task "Manager Zone Selection", which determines the zone responsible for supervising the allocation of the requested SFC. Next, the task "SFC Segmentation" divides the SFC into segments, allowing the SFC to be executed in multiple domains. Subsequently, the task "Execution Zone Selection" assigns each segment to the appropriate zone. Task "VNFs Allocation" further specifies the node within the Compute Zone to execute each VNF. Finally, the task "Network Path Building" coordinates the establishment of virtual links between the domains where the VNFs have been allocated.

4.1. Metadata Consolidation

In this section, we present how the metadata flows in the topology according to the proposed solution. This process generates data used by the Manager Zone Selection and Execution Zone Selection tasks. The Aggregation Zones process and store the metadata sent by the children's zones. By adopting metadata consolidation strategies, the Aggregation Zone will provide partial information to its parent zone. The consolidated metadata in each Aggregation Zone is created over i) the metadata sent by the children Aggregate and Compute zones, ii) the metadata already stored in the Aggregation zone from previous interactions with the children zones, and iii) metadata gathered by the Aggregation zone, like the network status of the Aggregation zone. The children zone will provide to the parent zone raw metadata composed of i) zone name, ii) VNF Types that can be executed, iii) zone border gateway, iv) cost to execute the VNF Type, and v) delay between children and parent zones. The Aggregation Zone updates the consolidated metadata in response to changes reported by the children zones. When the consolidated metadata is updated, the Aggregation Zone will inform the parent zone about its new state.

The children zone will send metadata to the parent zone when the zone is configured in the environment, or the metadata changes by the following situations: i) when a previously available VNF Type can no longer be executed, ii) when a VNF Type that

was previously unavailable becomes available, due to the release of resources that were previously allocated, and iii) when a new VNF Type becomes available. The information about the available resources in each physical node inside a Compute Zone will not be sent to the parent zone. Retaining node resources and topology information ensures data privacy, thus improving infrastructure security [Xu et al. 2018].

Fig. 1 depicts the consolidation of metadata over time. In time $T0$, $A2$ stores the information that the best Compute Zone to execute a "VNF Type 4" is zone $C2$. At time $T1$, $C2$ informs $A2$ that "VNF Type 4" cannot be executed anymore; therefore, $A2$ updates its consolidated metadata, removing the possibility of "VNF Type 4" being executed because none of the zones of children $A2$ can execute this VNF Type anymore. In time $T2$, the zone $C1$ and $C2$ will inform $A2$ that they can execute "VNF Type 4", and $A2$ updates the consolidated metadata, indicating that zone $C1$ is now the adequate zone to execute "VNF Type 4" once the delay from $C1$ to border gateway 1 is lower than $C2$.

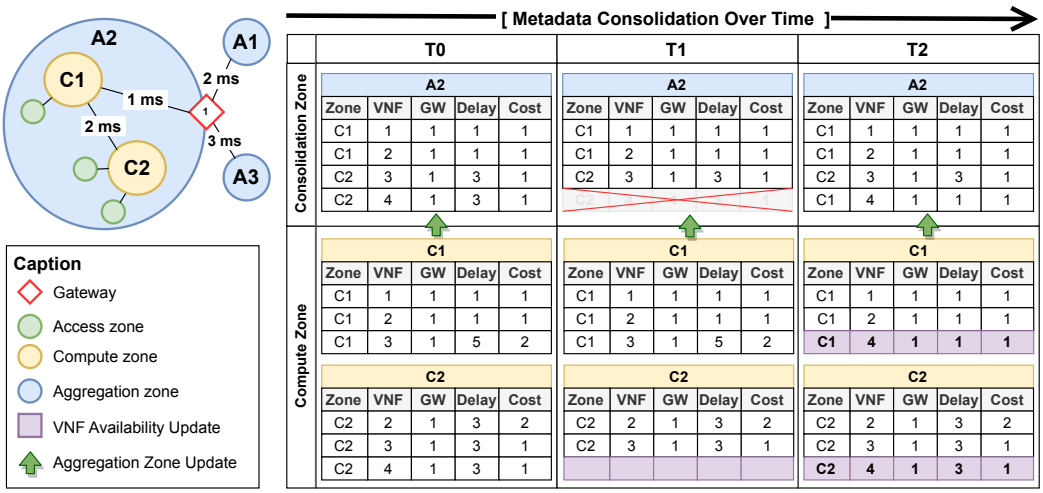


Figure 1. Metadata consolidation in a distributed environment over time

4.2. Players Strategy Update

We modeled the SFC placement problem as a Singleton Congestion Game. A key aspect of this class of games is how each player selects its strategy, which corresponds to choosing the zone where its VNFs will be executed. In this section, we detail the process by which players update their strategies during the execution of the game.

We adopt an approach based on the Best-Response Dynamics (BRD) method [Swenson et al. 2018]. In BRD approaches, each player iteratively updates their strategies by selecting the best response to the current strategies of the other players. This technique is commonly used in SGC to model the behavior of rational players who iteratively improve their own outcomes. The goal is to reach a stable state where no player can reduce its cost by independently changing its strategy. Fig. 2 illustrates the activity diagram that depicts the execution of the strategy based on the BRD approach.

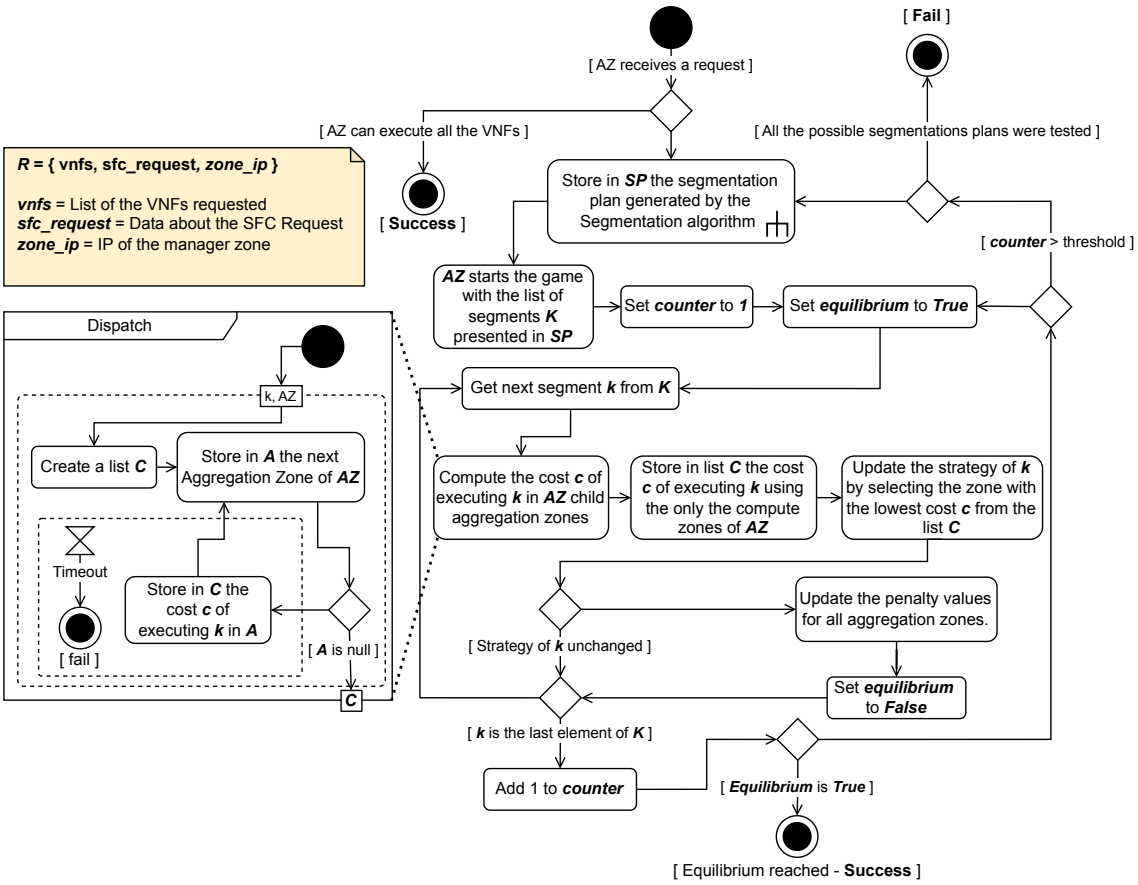


Figure 2. Players strategy update activity diagram.

5. Performance Evaluation

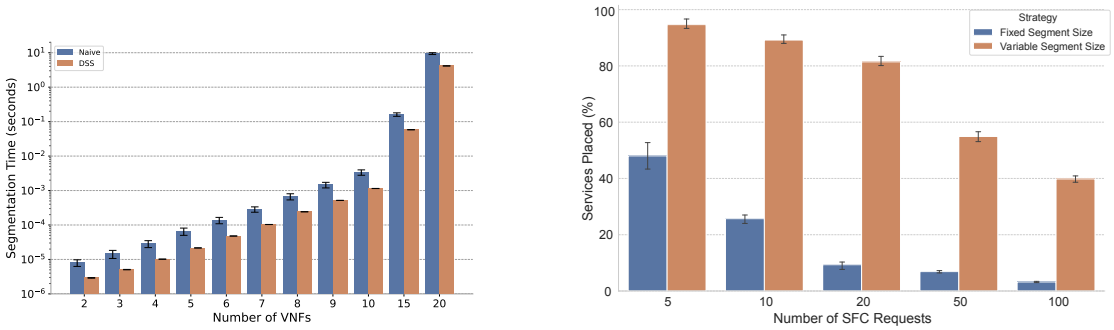
In this section, we describe the execution of our proposed solution in a simulated environment. The source code of the implementation and data is available on GitHub [1]. The simulation was defined based on [Chen et al. 2022]. The topology was the Internet Topology Zoo [Knight et al. 2011], which was used to organize the distributed environment. This topology has 64 nodes. Each node in the topology was considered a domain. We create scenarios with 5 to 100 SFC requests. The SFC requests arrive following a Poisson distribution. The CPU and memory requirements for each VNF Type were obtained from the respective website. The simulations were executed 10 times for each scenario, and the results are reported as mean values. We computed the 95% confidence interval assuming a normal distribution.

5.1. SFC Segmentation

In this section we present experiments regarding the proposed SFC Segmentation approach. The independent variables are i) the *Number of VNFs*, that indicates how many VNFs are in the SFC, and ii) the *Number of SFC Requests*, showing the number of SFCs that were requested by the users. The dependent variables are i) the *Segmentation time*, indicating how long the process took to create the segmentation plans, and ii) the *Service Placed*, that measures the percentage of SFCs placed.

[1] `https://github.com/anselmobattisti/speed`

Figure 3a shows the average time required to generate the segmentation plan as the length of the SFC increases. The y-axis indicates the average generation time on a logarithmic scale. We compare our approach with a naive method that exhaustively generates all possible VNF combinations. The results demonstrate that, using our approach, the time required to generate the placement plan for an SFC with 20 VNFs is less than 10 ms, thus not impacting the total SFC placement time.



(a) Segmentation Time by SFC Size **(b) SFC Placement Success Rates**

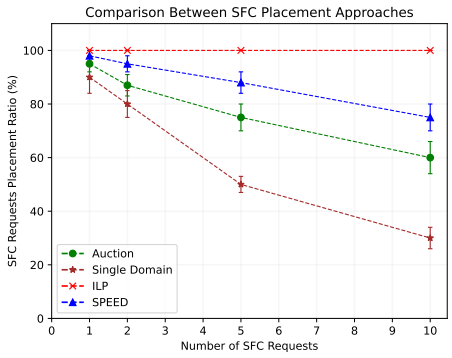
Figure 3. Evaluation of SFC segmentation regarding time and success rate

Figure 3b illustrates the SFC placement success rates achieved by **SPEED** using two segmentation strategies: (i) *Fixed Segment Size*, where each segment contains the same number of VNFs, and (ii) *Variable Segment Size*, where segment sizes vary according to the characteristics of the environment. The x-axis shows the number of SFC requests $[5, 10, 20, 50, 100]$, while the y-axis displays the percentage of successfully placed SFCs. The results demonstrate that the variable segment size strategy leads to higher placement success rates, notably as the number of requests increases.

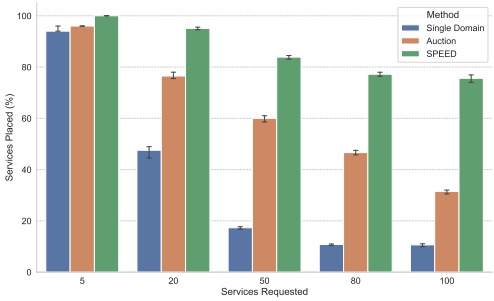
5.2. Distributed SFC Placement

This section presents the performance evaluation for SPEED. We compare our work against a Single Domain, an Auction, and an Integer Linear Programming approach. We adopt the metrics: i) *SFC Placement Success Rate*, which is the ratio between the number of SFCs placed and the number of SFCs requested, and ii) *SFC average execution cost*, which is the monetary cost to execute the requested SFC. Due to the size of the problem, it was only feasible to compare the ILP method in the first experiment.

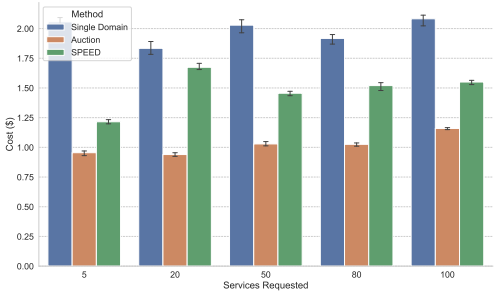
Fig. 4a compares the methods based on the SFC placement success rate metric. To carry out this experiment with the ILP, a small scenario with only five compute nodes was created. In the ILP approach, all requests were successfully placed. In contrast, the SPEED approach exhibited a success rate 20% lower than the ILP when handling 10 requests. The SPEED method, despite its lower success rate in this context, brings the benefits of enhanced scalability and fault tolerance that are often constrained in centralized solutions like those assumed in ILP-based and single domain approaches. Comparing SPEED and the auction approach, we can notice that SPEED performs better, which can be attributed to the better segmentation process adopted. Therefore, while the ILP method shows superior performance in controlled small-scale scenarios, the practical applicability of the SPEED approach in complex networks will be presented in next experiments.



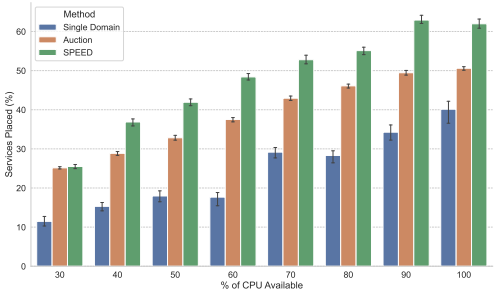
(a) Comparison of Placement Methods



(b) SFC Placement Success Rate



(c) SFC Average Execution Cost



(d) Success Rate by CPU Availability

Figure 4. Evaluation of SFC placement and execution cost

In Fig. 4b, we increase the number of SFC Requests to test the scalability of our proposed solution and compare it with the different SFC placement methods with respect to their SFC placement success rate metric. The SPEED approach can split the execution of the requested VNFs that encompass an SFC into multiple domains. Therefore, the number of suitable execution plans increases compared to approaches that allocate all VNFs to the same domain. SPEED also exhibits a more gradual decline in the number of SFCs executed. This characteristic implies that in resource-scarce scenarios, our approach has a higher success rate. For example, in the scenario with 100 SFC requests, we found a suitable plan for more than 75% of them, and compared to the auction approach, we increased the number of plans successfully placed by 30% on average.

The choice of resources for executing the SFC significantly impacts the financial cost of the operation. Consequently, the selection of the appropriate resources is a key factor in the profitability of service providers. Fig. 4c compares the average cost to execute each requested SFC. The results show that even with a higher success placement rate, the average cost to execute each SFC was similar to that of previous approaches. Therefore, our proposed approach, despite increasing the number of requests placed, maintains similar costs in the execution of each SFC.

In multi-domain environment, the resources of a zone were not exclusively reserved for the execution of SFCs; there are usually other systems that also utilize resources in the same zone. Fig. 4d illustrates a scenario in which the number of SFC requests was set to 50, while the resources available at the nodes to execute SFCs were restricted to a

percentage of their total resources. The results show that SPEED outperforms the other approaches in scenarios with scarce resources.

6. Conclusion

This work introduced SPEED, a distributed approach to the SFCPP that advances the state-of-the-art through a distributed segmentation process and game-theoretical modeling of resource allocation. Experimental results demonstrated that SPEED increased the service placement success rate by 30% compared to existing techniques, showing particular effectiveness in resource-constrained environments. Furthermore, the proposed method ensured that this improved performance is achieved while maintaining consistent execution costs for each service.

By minimizing inter-domain communication regarding topology and resource availability, SPEED effectively reduces overhead while preserving privacy in multi-provider environments. The integration of distributed segmentation and congestion game theory allows the framework to improve placement success rates and to scale across complex, real-world edge-cloud infrastructures. Consequently, this decentralized approach represents a significant contribution to the SFCPP literature.

Future work includes integrating SPEED into production-grade platforms like OSM or EMCO for real-world environments. We also intend to expand analyses against the latest state-of-the-art solutions, focusing on performance gains as network scale increases. Finally, the framework will be tested under diverse, complex topologies to further validate its robustness and statistical reliability.

During the development of this thesis, four papers were published in journals, three papers were presented in conferences, and six international patents were filed. The most relevant papers are i) [Battisti et al. 2024], which presents the distributed approach to solving the SFC Segmentation problem, and ii) [Battisti et al. 2026], which represents the final and most comprehensive consolidation of the doctoral research.

Acknowledgments: This work was supported by CAPES, CNPq, MCTI, FAPERJ, and INCT-ICoNIoT. Flavia C. Delicato and Débora Muchaluat-Saade are CNPq and FAPERJ Fellows.

References

Avasalcai, C., Tsigkanos, C., and Dustdar, S. (2019). Decentralized Resource Auctioning for Latency-Sensitive Edge Computing. In *2019 IEEE EDGE*, pages 72–76.

Battisti, A. L. É., Delicato, F. C., and Muchaluat-Saade, D. C. (2024). A novel method for sfc segmentation in edge-cloud environments. In *2024 IEEE CloudNet*, pages 1–8.

Battisti, A. L. É., Muchaluat-Saade, D. C., and Delicato, F. C. (2026). SPEED: A Distributed Approach for SFC Placement in the Edge-Cloud Continuum. *Annals of Telecommunications*, 9.

Bhamare, D., Jain, R., Samaka, M., and Erbad, A. (2016). A survey on service function chaining. *Journal of Network and Computer Applications*, 75:138–155.

Chen, C., Nagel, L., Cui, L., and Tso, F. P. (2022). Distributed federated service chaining: A scalable and cost-aware approach for multi-domain networks. *Computer Networks*, 212:109044.

Kapassa, E., Touloupou, M., and Kyriazis, D. (2018). SLAs in 5G: A Complete Framework Facilitating VNF and NS Tailored SLAs Management. In *IEEE WAINA*, volume 1, pages 469–474.

Kaur, K., Mangat, V., and Kumar, K. (2020). A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture. *Computer Science Review*, 38:100298.

Knight, S., Nguyen, H. X., Falkner, N., Bowden, R., and Roughan, M. (2011). The internet topology zoo. *IEEE JSAC*, 29(9):1765–1775.

Liu, Y., Zhang, H., Chang, D., and Hu, H. (2020). Gdm: A general distributed method for cross-domain service function chain embedding. *IEEE Transactions on Network and Service Management*, 17(3):1446–1459.

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358.

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., and Boutaba, R. (2016). Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262.

Mostafavi, S. and Hakami, V. (2021). Quality of service provisioning in network function virtualization: a survey. *Computing*, 103(5):917–991.

Pan, J. and McElhannon, J. (2018). Future Edge Cloud and Edge Computing for Internet of Things Applications. *IEEE Internet of Things Journal*, 5(1):439–449.

Santos, G. L., Bezerra, D. d. F., Rocha, É. d. S., Ferreira, L., Moreira, A. L. C., Gonçalves, G. E., Marquezini, M. V., Recse, Á., Mehta, A., Kelner, J., Sadok, D., and Endo, P. T. (2022). Service Function Chain Placement in Distributed Scenarios: A Systematic Review. *Journal of Network and Systems Management*, 30(1):4.

Sun, G., Li, Y., Liao, D., and Chang, V. (2018). Service function chain orchestration across multiple domains: A full mesh aggregation approach. *IEEE TNSM*, 15(3).

Swenson, B., Murray, R., and Kar, S. (2018). On Best-Response Dynamics in Potential Games. *SIAM Journal on Control and Optimization*, 56(4):2734–2767.

Wang, S., Cao, H., and Yang, L. (2020). A Survey of Service Function Chains Orchestration in Data Center Networks. In *2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6, Taipei, Taiwan. 2020 IEEE Globecom Workshops (GC Wkshps, IEEE.

Xu, Q., Gao, D., Li, T., and Zhang, H. (2018). Low latency security function chain embedding across multiple domains. *IEEE Access*, 6:14474–14484.

Yi, B., Wang, X., Li, K., k. Das, S., and Huang, M. (2018). A comprehensive survey of Network Function Virtualization. *Computer Networks*, 133:212–262.