Annals of Telecommunications
https://doi.org/10.1007/s12243-026-01157-w

Speed: a distributed approach for SFC placement in the edge-cloud continuum

Anselmo Luiz Éden Battisti[1] · **Flavia C. Delicato**[1] · **Débora Christina Muchaluat-Saade**[1]

Received: 11 September 2024 / Accepted: 23 January 2026

Abstract
In the last decades, there has been a trend to virtualize computing and networking resources over the physical infrastructure. This approach was initially adopted in the core functions of the network, thus creating the network function virtualization paradigm. The process which defines the computational nodes and links to execute a set of virtual network functions (VNFs) 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 where the VNF instances will be executed is required. In this paper, we propose a new approach, named SPEED, to solve the SFC placement problem (SFCPP) 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 of a simulation-based performance evaluation show that SPEED is feasible and, compared to other approaches, has a gain of 30% in the number of SFC requests placed successfully.

Keywords SFC placement · Distributed SFC placement · VNF placement

1 Introduction

In recent decades, there has been a trend to virtualize computing and networking resources over physical infrastructure [1]. This technology enabled the development of multiple computing environments such as cloud, edge, Internet of Things (IoT), and 5G [2]. 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 [3]. NFV decouples network functions from the underlying dedicated hardware and executes them through software, which is called virtual network function (VNF) [4].

Today, even high-level functions directly requested by users, such as video encoders and antivirus, are likewise executed as VNFs [5]. This technology has gained momentum because the adoption of the NFV paradigm reduces the space needed for network hardware, decreases network power consumption, decreases network maintenance costs, increases the life cycle of network hardware, and increases resource provisioning speed and efficiency, facilitating the management and orchestration of network functions [1, 3, 4].

In traditional environments, the network functions were executed by physical devices, and the network team had to manually define the local of the physical network where the appliance hardware was installed. However, with the adoption of the NFV paradigm, this task must be automated. This has led to a new problem known as VNF Placement, which defines the computational node to execute each VNF [6].

Flavia C. Delicato and Débora Christina Muchaluat-Saade contributed equally to this work.

✉ Anselmo Luiz Éden Battisti
anselmo@midiacom.uff.br

Flavia C. Delicato
fdelicato@ic.uff.br

Débora Christina Muchaluat-Saade
debora@midiacom.uff.br

[1] Computer Science Department, MídiaCom Lab, Institute of Computing, Av. Gal. Milton Tavares de Souza, s/nº São Domingos, Niterói 24210-346, RJ, Brazil

Published online: 21 February 2026

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 [7], creating a new concept called service function chain (SFC) [8]. An SFC is a chain of sorted VNFs, associated with a service level agreement (SLA) [9]. 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 [10]. 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 [11]. The SFCPP can be described as *giving a requested SFC, a set of conditions, restrictions, and the 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* [5]. The SFCPP has already been proven to be an NP-Hard problem [12], so creating good heuristics is fundamental to addressing it.

With the advance of edge-cloud continuum, small- to medium-sized compute nodes are now available that aim to provide additional computing, storage, and networking resources to applications deployed across the continuum Internet of Things (IoT) devices, edge, and cloud [13]. These nodes are typically operated by several owners, forming a multi-domain environment [14]. Thus, these compute nodes can cooperate with each other to meet the needs of a complete SFC. Therefore, the VNFs of an SFC can be deployed and executed in nodes in different domains, potentially managed by different service providers [15].

Most approaches to solve SFCPP in multi-domain scenarios rely on a centralized approach. However, this strategy has shown several limitations, as highlighted in previous studies [5, 16–19]. Some of the most notable limitations are as follows:

- **Scalability issues:** As the number of compute and network nodes increases, maintaining a complete view of the infrastructure becomes computationally prohibitive.
- **High latency:** Centralized orchestration increases the time required to compute placement plans, which is incompatible with latency-sensitive services.
- **Confidentiality concerns:** Domains are typically owned by different providers who are reluctant to expose internal topologies, resource availability, or cost models.
- **Single point of failure:** Centralized orchestrators introduce a vulnerability—if the central node fails, the entire placement process stops.
- **Limited fault tolerance:** A centralized solution has lower resilience in dynamic environments where nodes or links may fail frequently.
- **Inefficient use of local knowledge:** Local domains may have up-to-date information about their state that is not fully captured or promptly propagated to a central orchestrator.
- **Poor adaptability:** Centralized systems often struggle to react quickly to changes in resource availability, load, and topology updates.
- **Regulatory and legal constraints:** In some scenarios, data jurisdiction laws prevent infrastructure providers from sharing certain types of data across domains or borders.

Compared to solutions proposed in the literature, which rely on static segmentation, aggregated graphs, or distributed auctions with high overhead, SPEED offers a novel strategy for executing the SFC placement in distributed environments based on a distributed segmentation and congestion game theory. This combination provides better adaptability in multi-domain environments. The contributions of this work are as follows:

1. A novel approach to solving the SFC placement problem in a multi-domain environment through a distributed method using singleton congestion game.
2. A new strategy for dynamically segmenting the VNFs of the SFC, enabling execution across different domains.
3. 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.

This paper is organized as follows. In Section 2, we review related work and background. Section 3 presents our SFCPP system model. Our proposed framework, named SPEED, is detailed in Section 4. The experimental results and discussion are provided in Section 5. Finally, Section 6 concludes this work and highlights future perspectives.

2 Background/related work

2.1 Game theory

In this section, we introduce concepts of game theory and its relation with the SFCPP. Game theory is a framework for designing interactions among players who desire to achieve some goals [20]. The focus of game theory is to provide tools for analyzing scenarios in which players make interdependent decisions to achieve their goals [21].

Multiple classes of problems can be modeled using game theory [21]. One type of game was proposed by Rosenthal in 1973 [22] named the congestion game. In a congestion game, each player's outcomes will be influenced by (i) the resources they choose and (ii) the number of players that choose the same resource. Equation 1 formally describes a congestion game:

$$\Gamma = (N, R, (\Sigma_i)_{i\in[n]}, (d_r)_{r\in[R]}) \quad (1)$$

The elements of the Eq. 1 are defined as follows:

- A finite set of players $N = (1, 2, ..., n)$, where $|N| = n$.
- A finite set of resources $R = (1, 2, ..., r)$, where $|R| = m$, these resources can be modeled as the edges of a directed weighted graph, for example.
- Each player $n \in N$ has a set of strategies that it can play named $\Sigma_i \subseteq 2^R$, where 2^R is the powerset of all possible strategies.
- Each element $r \in R$ has a cost function $d_r : \mathbb{N} \rightarrow \mathbb{R}$, where $\mathbb{N}$ is the number of players using the resource and, $\mathbb{R}$ is the total cost.
- The cost for player i, to adopt a finite set of strategies $S = (s_1, s_2, ..., s_n)$, is $c_i(S) = \sum_{r\in S_i} d_r(n_r(S))$, where $n_r(S) = |i \in N|$ and $r \in S_i$. That is, the sum of the cost of multiple payers adopting the same strategy.

All the congestion games have a Nash equilibrium (NE) according to [22]. NE is a state where no player can improve its outcomes by unilaterally changing one of its strategies, and the other players maintain their strategies [23, 24]. The computational cost to reach the NE in congestion games is n^m, where n is the number of players and m is the number of resources. Reaching the NE in an environment with many players and multiple resources is an algorithmically complex [24].

The computational effort required to reach the NE grows with the scale of the system, particularly as the number of available resources and participating players increases. However, there is a subset of the congestion game named singleton congestion game (SCG) [24], with lower computational complexity [23]. A congestion game is called singleton if, for every player $i \in N$ and every $R \in \Gamma_i$, it holds that $|R| = 1$. In other words, all players must select a single resource from a subset of allowed resources [25].

Modeling the SFC placement problems in a distributed environment using game theory has not been proposed yet. Few works utilize game theory to model other problems related to SFCs. In particular, the authors of [26] investigate whether better performance is possible when restricted to the load balancing problem in a game where players can only choose a single resource.

2.2 SFC placement problems

In this section, we present some relevant proposals that deal with the SFC placement problem (SFCPP). This problem can be addressed in a centralized or distributed fashion. Centralized approaches are the most common, and some examples of solutions are depicted in Table 1. They have a unique compo-

Table 1 Centralized SFC placement approaches

Study	Objective	Technique	Environment
Ruiz et al. [16]	Mono	Genetic algorithm	Cloud
Kim et al. [17]	Mono	Conventional Lightchain algorithm	N/D
Emu et al. [18]	Mono	Machine learning	Edge
Garrich et al. [27]	Mono	Greed	Edge
Li et al. [28]	Mono	Greed	Edge
Nguyen et al. [29]	Multi	Markov chain	Edge and cloud
Reyhanian et al. [30]	Multi	Alternating direction method of multipliers	Edge
Kiran et al. [31]	Multi	Genetic algorithm	Edge
Pei et al. [32]	Multi	Machine learning	N/D
Bunyakitan et al. [33]	Multi	Q-learning scheme	Edge
Pham et al. [34]	Multi	Markov chain	Cloud
Li et al. [35]	Multi	Deep reinforcement learning	Cloud
Niu et al. [36]	Multi	Graph-based particle swarm optimization	Edge
Alahmad et al. [37]	Multi	Greed	Cloud
Gao et al. [38]	Multi	Steiner tree and Markov decision	Cloud
Mohamad et al. [39]	Multi	Greed	Edge
Battisti et al. [14]	Multi	Greedy with support for VNF instance sharing	Edge and cloud

nent with full knowledge of the network topology, domains, node resources, and capabilities. Some drawbacks of centralized approaches are (i) lack of scalability, (ii) network communication cost to gather data about all the resources, and (iii) privacy and security in a multi-provider environment [5].

To overcome the issues related to centralized approaches, distributed approaches began to be proposed. In a systematic review, Santos et al. [5] present that 60% of the works about decentralized SFC placement were published after 2019. In fact, the main platforms adopted by SFC providers, like Open Source MANO (OSM) or Open RAN (O-RAN), do not offer any distributed SFC Placement solutions [40, 41]. Distributed approaches to solve the SFCPP were less common than centralized ones.

Chen et al. [42] present a distributed SFC Placement in a multi-domain environment where each domain is independent, shares minimum information about the internal infrastructure, and has its own orchestrator. The SFC request arrives in any domain, named ingress domain, and the orchestrator of the ingress domain coordinates the SFC Placement. The orchestrator of the ingress domain builds an aggregated graph including inter-domain links and aggregated nodes. Each domain is converted into one aggregated node, which stores the total CPU, available memory, and the cost of the resources in the domain. The ingress orchestrator employs the k-shortest path algorithm in the aggregated graph and decides how to assign the VNFs to the different domains. The authors call this process SFC Partition, and it consists of the distributed step of the algorithm. Subsequently, each partition is sent in parallel to the selected domains, and each orchestrator finds a solution (placement plan) within the resources of their own domain and sends deployment results back to the ingress orchestrator. If the process fails due to a lack of resources, the ingress orchestrator selects the next k-shortest path.

However, the work of [42] has some drawbacks. The strategy adopted to create the domain aggregated node implies that the resources of all the nodes have the same cost, which is not true in most realistic scenarios. Another point is to allocate VNFs in nodes that demand more energy, increasing the total cost of operation in the domain. Furthermore, if many nodes have few available resources, the aggregated information will lead the ingress domain to possibly select domains that actually do not have a node with the resources required to run a VNF. Moreover, the algorithm creates the aggregation graph for each SFC request, which is not feasible for regional-scale networks. They also consider that the link requirements between all VNFs are the same, which is not true in most cases. Finally, the segmentation phase does not use the fallback information to create a better segmentation plan, increasing the number of rounds to allocate an SFC completely.

Another approach was proposed by Liu et al. [43]. They present a distributed SFC placement and load balance 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. The SFC request can arrive in any domain, named ingress domain. The ingress domain orchestrator creates segments from the VNF that compose the SFC. At least one domain should have resources to handle the defined segment. After the segmentation phase, each segment is sent to candidate domains, and they use a distributed auction strategy to decide which domain will execute each segment.

Nonetheless, the work of [43] has some disadvantages. The adoption of a distributed auction strategy is typically employed in scenarios where the participants are competing against each other, in cooperative environments, like federations, where the objective is to maximize the general wealth rather than the individual gain. Another flaw is that the distributed auction [44] converges in 2Δ iterations without changes in the local winning list. As Δ is the domain-level network diameter, which is the number of inter-links included in the shortest path connecting the furthest domain pair, we can infer that in complex networks, the number of iterations can be considerable, making the adoption of that strategy infeasible. Finally, they point out that in the load balancing phase, only domains that do not participate in the auction compete, i.e., only domains that were considered not feasible to run the segment can compete to run the segment in the load balance, which is a counter-sense.

Gang et al. [45] 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 which VNFs it can execute. The partitioning phase consists of creating sub-SFCs. This partition can be customized to (i) minimize the number of domains or (ii) improve the load balance of consumed resources in the domains. The placement phase that each domain executes to place the sub-SFC can be customized to (i) minimize the delay or (ii) improve the load balance in the nodes of the domain. The cornerstone of their work resides in the full-mesh aggregation [46] approach for creating the topology of the aggregated graph. Therefore, in scenarios with a complex environment, this can consume valuable resources unnecessarily.

The approach proposed by [47] is based on a decentralized auction strategy. They aim at finding a deployment mapping for each VNF of an SFC compliant with the resource requirements and latency constraints, besides increasing the privacy of each domain. Each domain that participates in the auction sends a bid value for each VNF of the SFC. A centralized

component receives the bids and runs an algorithm based on the satisfiability modulo theories to match each VNF with each domain. However, the proposed approach rapidly increases the computational demand in environments with more than 20 nodes.

Multi-domain environments typically exhibit greater infrastructure scale and topological complexity compared to single-domain environments. Therefore, maintaining the topology information in a single node, which is commonly adopted in centralized approaches, faces challenges, such as storage capabilities and computational resources. Centralized approaches also pose business model challenges, such as the need for sharing private data about the network with external entities. These aspects make centralized approaches not suitable in multi-domain environments [5]. Storing partial information (consolidated metadata) in multiple zones is a strategy already adopted in other contexts, for example, to solve the hierarchical routing problem [46]. We claim that this approach can also be used to solve the SFCPP in a distributed multi-domain environment.

Another aspect that the algorithms presented in this section addressed, during the placement of the SFCs into a distributed environment, is how the SFCs will be segmented. The segmentation problem handles the split of the VNFs that compose the SFC into segments [48]. There are three possible ways to segment an SFC: (i) *No segmentation*, where all the VNFs are executed in the same domain; (ii) *Static*, where the segmentation is defined in the beginning of the placement process and does not change during the next steps of the placement; and (iii) *Dynamic*, where the segmentation process is defined dynamically during the creation of the placement plan. Table 2 compares our proposal with the literature.

3 System model

In this section, we present the system model considered in our work for the SFC placement problem. The problem was elaborated as a mixed-integer programming (MIP) model. We adopt the minimization of total placement cost as the objective function. This cost can encompass different aspects depending on the deployment context, such as CPU and memory usage, energy consumption, or financial expenses charged by infrastructure providers. This choice reflects real-world priorities in multi-domain environments, where cost efficiency plays a central role in SFC deployment strategies. Table 3 lists the key notation.

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 $Z = \{z_1, z_2, ..., z_n\}$ set 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 $\Re = \{r_1, r_2, ..., r_n\}$ set denotes all SFC requests. Requests arrive at the system either via an edge or a cloud node. 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 is defined in Eq. 2, 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 $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$ represents the network zones. Each zone $\omega_i \in \Omega$ has the following 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 $parent ==$

Table 2 Distributed SFC placement approaches

Study	Technique	Environment	Segmentation	Orchestrator
Chen et al. [42]	Graph theory	Edge/cloud	Static	Distributed
Liu et al. [43]	Auction	Edge/cloud	Static	Distributed
Gang et al. [45]	Mesh aggregation	Edge/cloud	Static	Centralized
Avasalcai et al. [47]	Auction	Edge	Static	Centralized
SPEED	**Game Theory**	**Edge / Cloud**	**Dynamic**	**Distributed**

Table 3 System model notation

Notation	Description
$G = (\Omega, L)$	Undirected graph of the physical network
Ω	Set of network zones
L	Physical links between the zones
Z	Set of all the VNFs in the environment
$\Re$	Set of SFC Requests
r	SFC Request definition
src	Source node of the dataflow node
dst	Destination node of the dataflow
V	VFN that composes the requested SFC
VL	Virtual Links that bind the VNFs
vl	Virtual Link that connects two VNFs
max_delay	Maximum delay between two VNFs
$childrens_\omega$	Children's of zone ω
$parent_\omega$	Parent zone of ω
τ	Time when the SFC Request arrives
Λ_ω	VNFs that can be executed by at least one zone ω subordinate
cpu_z	CPU required to run a VNF
mem_z	Memory required to run a VNF
bw	Required network bandwidth
max_delay	Maximum tolerable delay of an SFC
ω	Defines a single zone
Φ	Whether a VNF is permitted or prohibited for deployment in a zone
κ_ω	Type of zone, binary variable (Compute Zone = 1, otherwise 0)
$CompCost$	Computing cost to execute the VNFs
$NetCost$	Cost of the network to handle the traffic
Decision variables	
x_{ij}	If VNF n_i will be executed in zone ω_j
y_{ij}	If virtual link v_i will be executed in link l_j

$NULL$ it means that the zone is the highest element in the hierarchy, $type_\omega$ identifies whether it is an aggregated or a Compute Zone. Equation 3 depicts the κ_ω that defines if it is a compute or an aggregate 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 $L = \{l_1, l_2, ..., l_n\}$ represents the links between the zones. For every link $l_i \in L$, between two zones ω_1 and ω_2, we use bw_l and $delay_l$ to denote its bandwidth capacity and delay, respectively. The parameter $\Psi_{l_j}^{vl_i}$ represents the cost of executing the virtual link vl_i in the link l_j.

$$\Phi_{\omega_i,\omega_j,z} = \begin{cases} 1, & \omega_i \text{ allows VNF } z \text{ in } \omega_j \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$$\kappa_\omega = \begin{cases} 1, & ComputeZone \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The SFC execution consumes computational and networking resources, for which the service provider will charge a fee to execute the SFC requested. Therefore, our problem formulation requires the decision variable x_{ij} that defines whether VNF z_i will be executed in zone ω_j and the decision variable y_{ij} that defines whether virtual link v_i will be executed in link l_j.

Equation 4 defines how the cost of computational resource is calculated. Equation 5 defines how the cost of network resources is calculated. Thus, our objective is to minimize the total placement cost, defined in Eq. 6.

$$CompCost = \sum_{z_i \in Z} \sum_{\omega_j \in \Omega} x_{ij} * \beta_{\omega_j}^{z_i} \tag{4}$$

$$NetCost = \sum_{v_i \in V} \sum_{l_j \in L} y_{ij} * \Psi_{l_j}^{v_i} \tag{5}$$

$$minimize\ CompCost + NetCost \tag{6}$$

The problem is subject to some constraints. Equation 7 defines that the VNF can only be executed in compute zones. Equation 8 defines that the VNF can only be executed in zone ω when the VNF is available in that zone. Equation 9 defines that the VNF can only be executed in zone ω when there are no affinity (Φ) restrictions. These equations are constraints that enforce the existence of at least one valid placement option based on zone type, VNF availability, and affinity. They do not require all elements in the summation to satisfy the condition individually, but rather that the sum indicates at least one feasible placement.

$$\sum_{z_i \in Z} \sum_{\omega_j \in \Omega} x_{ij} * \kappa_{\omega_j} \geq 1 \tag{7}$$

$$\sum_{z_i \in Z} \sum_{\omega_j \in \Omega} x_{ij} * \Lambda_{\omega_j}^{z_i} \geq 1 \tag{8}$$

$$\sum_{z_i \in Z} \sum_{\omega_j \in \Omega} \sum_{\omega_k \in \Omega} x_{ij} * \Phi_{\omega_j, \omega_k, x_{ij}} \geq 1 \tag{9}$$

4 Proposed solution

This section presents the SPEED approach, which is a solution that addresses the SFC placement problem in a distributed manner. 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.

We assume that SPEED 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; zones are freely created by the service providers based on criteria like geolocation and resource capacity. Grouping the resources into zones has the following benefits: (i) helps enhance infrastructure security through data aggregation, (ii) allows the execution of SFC placement based on partial infrastructure information, and (iii) improves the scalability of the SFC placement in a complex environment.

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.

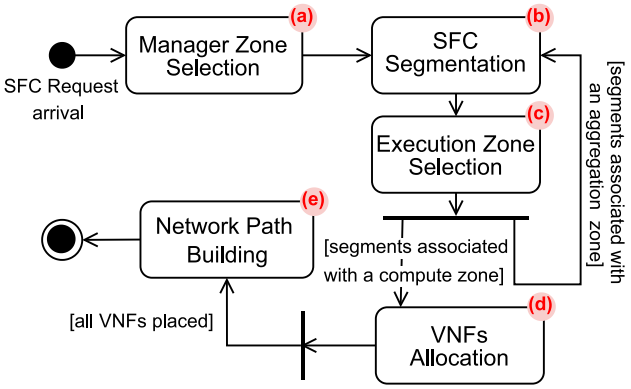


Fig. 1 SPEED workflow

Figure 1 shows the workflow of our proposed algorithm, providing an overview of the SPEED operation. It 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 Environment entities

This section presents an overview of the environment considered in the proposal. Figure 2 depicts an abstract view that encompasses (i) entities, (ii) zone types, and (iii) hierarchical topology. The access zones provide network connectivity for users; in a 5G scenario, these zones refer to the radio access network (RAN). The compute zones are nodes that offer computational resources to execute the VNFs; these nodes can represent either edge or cloud hosts. Lastly, the aggregation zones consolidate the metadata about either compute or aggregation zones.

All zones are associated with one aggregation zone, except the root zone. Thus, the elements of the distributed environment are arranged in a tree-like topology. Each level determines how the metadata is shared. The compute and aggregation zones send their metadata only to the aggregation zone to which they belong. The metadata flows from

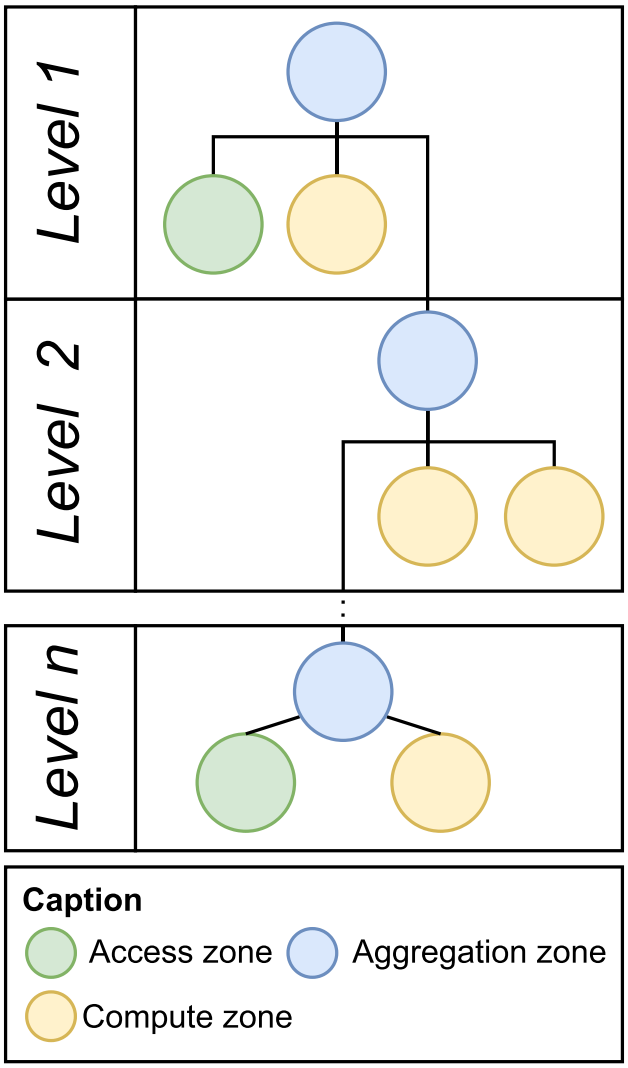


Fig. 2 Hierarchy organization

the bottom to the top-level zones. The top-level aggregation zones have consolidated information about their own compute zones and all the bottom-level aggregation zones.

4.2 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, depicted in Fig. 1a and b steps, respectively. 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. Furthermore, the aggregation zone can also update the consolidated metadata when a children zone fails to execute the requested VNFs. 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 [49].

Figure 3 depicts an environment and 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 $C2$.

4.3 Segmentation strategy

This section presents our strategy to solve the segmentation problem. Different segmentation plans can be created from the same SFC. The number of possible segment plans for an SFC is 2^{n-1}, where n is the number of VNFs composing the SFC.

The placement of segments in a multi-domain environment is a time-consuming task. Thus, adopting a strategy to choose a suitable segmentation plan regarding the environment is necessary to reduce the time to place the requested SFC [43]. In the literature, segments are typically defined during the initialization of the placement. Our approach, in

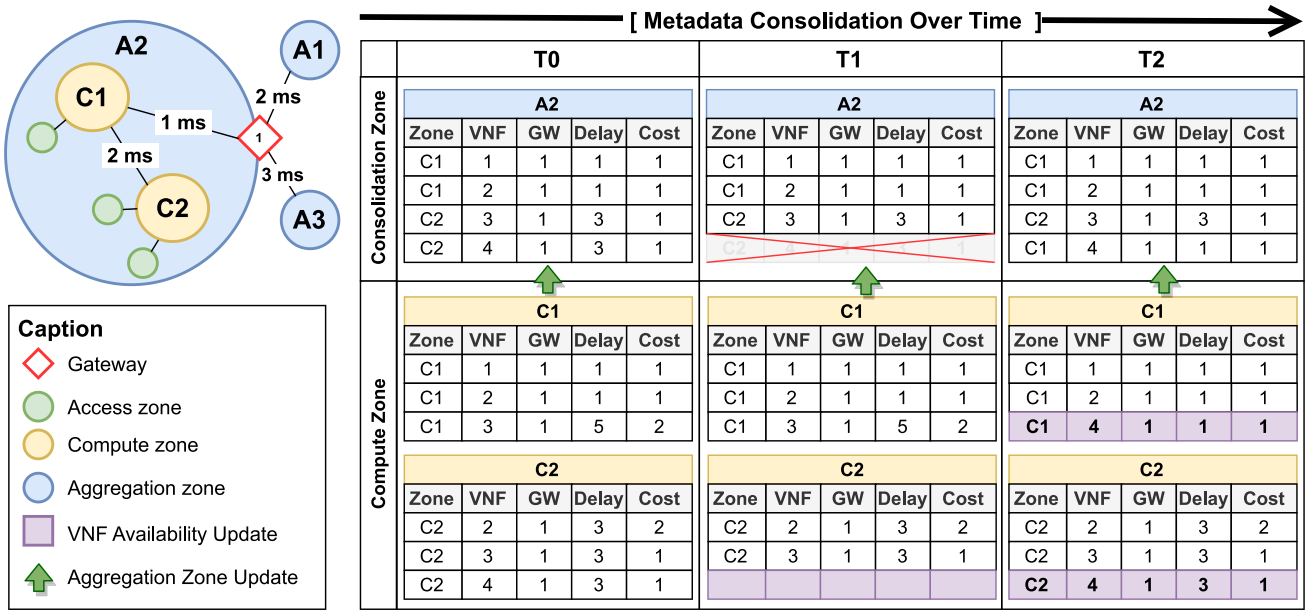


Fig. 3 Metadata consolidation in a distributed environment over time

contrast, defines the segments recursively, in the aggregation zone, multiple times, during the creation of the placement plan.

The problem of combining VNFs into segments can be mapped to a combinatorial problem similar to "stars and bars" [50]. "Stars and bars" can be used to solve counting problems, such as how many ways to put n indistinguishable balls into k distinguishable bins. The original problem does not maintain the order of the elements; however, the VNFs are distinguishable, which means that the position of each VNF in the SFC must be preserved during the segmentation phase. Therefore, we adapt "stars and bars" to ensure that the VNF chain remains organized to meet our problem requirements.

The authors in [43] define that a valid segmentation plan must comply with the conditions: (i) Each segment must contain at least one VNF; (ii) each VNF should be allocated within one segment; (iii) the VNF chain order in the SFC request must be maintained inside each segment; (iv) the order of segments must follow the VNF chain of the SFC request. We add a condition: (v) The number of segments must be smaller than the number of available children zones. This applies to our proposal, as plans with more segments than child zones are invalid since each segment must be executed in a distinct zone.

Our objective during the segmentation phase is to minimize bandwidth consumption between zones. By adopting this approach, we reduce the data transmitted between service providers. Therefore, during the segmentation phase, we seek to create a segmentation plan that not only complies with the SLA and minimizes the number of segments, facilitating the placement of the SFC in fewer zones. Additionally, we leverage VNFs that generate minimal output data as points to split an SFC.

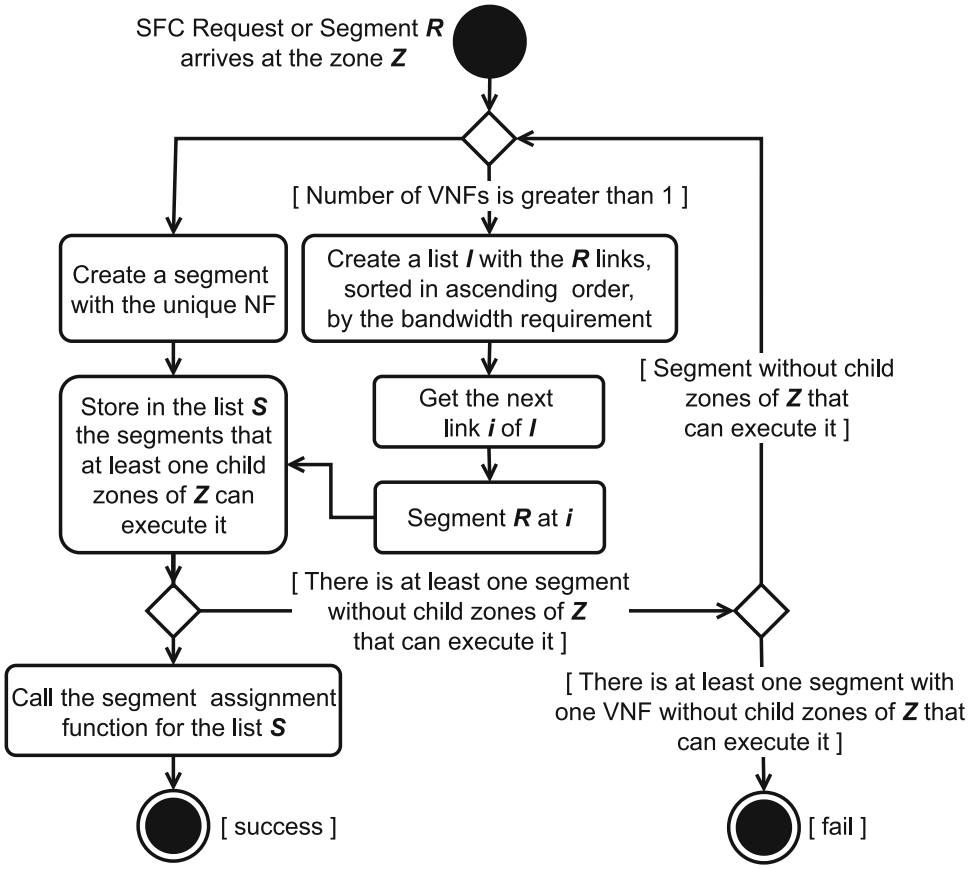
Figure 1b step illustrates where in our proposed solution the segmentation process takes place. Figure 4 depicts our heuristic to create and select the segmentation plan. In the proposed solution, whenever a plan is unfeasible, the previously created segments are split into new ones. If all segments have only one VNF and the plan fails, the segmentation phase will be considered to have failed. Another failure situation is when no children zone can handle any segment.

4.4 Heuristic for zone selection

In Section 2.1, we defined the main concepts of game theory, congestion games, and singleton congestion game (SCG). In this section, we present our heuristic that maps the problem of associating each segment to a zone and the SCG method. We mapped the SFC placement problem (SFCPP) into an SCG as follows:

- The set of players N is composed of segments.
- The set of resources R is composed of the bind between the Aggregation and its child zones.
- The set of strategies that each player $n \in N$ is the graph's edges that connect the *Aggregation zone* with the child zones.
- The cost function is the cost of executing the segment into a zone.

Fig. 4 Segmentation heuristic



Algorithm 1 depicts how the SGC decides which child zone should be selected to execute each segment. Each segment s can be associated with a different child zone. During the same game, the zone selected for the previous segment is used as input to select the next zone, thus increasing the chance of creating a feasible placement plan. The algorithm stops when all segments do not change the selected child zone based on the zones selected by the other segments.

The zones selected by Algorithm 1 represent a near-optimal solution. The algorithm is not recursive in its internal structure. However, it is executed recursively across hierarchical zones, where each zone manager applies the selection process to its subordinate zones.

Figure 5 portrays an example of how our proposed approach can solve the SFCPP in a distributed fashion. The topology comprises 1 access zone (`R1`), 8 compute zones, named `C1` to `C8`, and 5 aggregation zones, named `A1` to `A5`. Zones `A2`, `A3`, and `A4` are children of `A1`. Zone `A3` has the compute zone `C3` and aggregation zone `A5` as child zones. For simplicity's sake, we omitted each aggregation zone's aggregated data. The description of how the aggregated data is computed can be found in Section 4.2.

The SFC request arrives in access zone `R1`. Thus, `R1` is the source of the packet flow, and the destination is zone `C4`. The aggregation zone that can reach both `R1` and `C4` is `A1`.

Algorithm 1 Zone Selection based in a Singleton Congestion Game Approach.

Input: $Segments$, $ParentZone$
Result: Segmentation plan
Initialization:
$selZone \leftarrow NULL$
$equilibrium \leftarrow False$
$prevZone_s \leftarrow False, \forall s \in Segments$
$childZones \leftarrow getChildZones(ParentZone)$
while $equilibrium = False$ **do**
  $changed \leftarrow False$
  for each segment s **in** $Segments$ **do**
    $left \leftarrow$ if $s = 0$ then $NULL$ else $prevZone_{s-1}$
    $right \leftarrow$ if $s = |Segments| - 1$ then $NULL$ else $prevZone_{s+1}$
    $selZone_s \leftarrow suitableZone(s, childZones, left, right)$
    if $selZone_s \neq prevZone_s$ **then**
      $changed \leftarrow True$
    end
    $prevZone_s \leftarrow selZone_s$
  end
  if $changed \neq False$ **then**
    $equilibrium \leftarrow True$
  end
end
return $selZone$

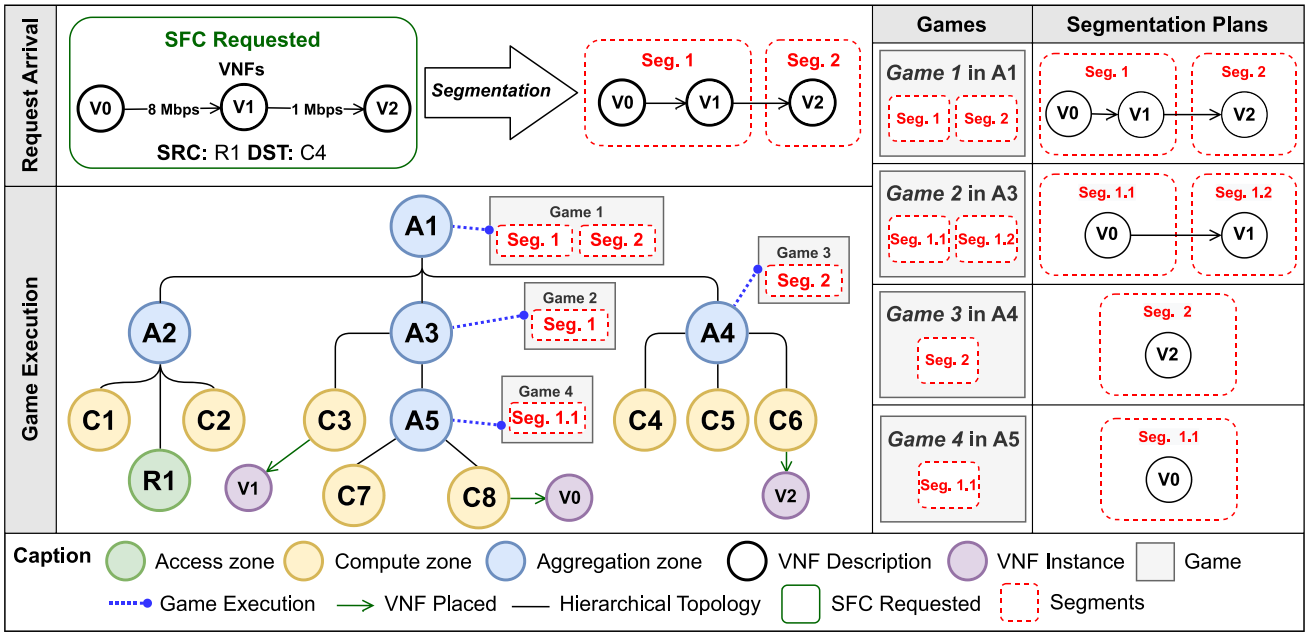


Fig. 5 Running example of our approach solving the SFCPP

Therefore, zone `A1` will be selected as the responsible for coordinating (coordination zone) the distributed process to allocate the requested service in the environment.

The first step of the process is to separate the VNFs of the SFC into segments. The requested service contains 3 VNFs, named `V0`, `V1`, and `V2`. The link between `V1` and `V2` requires a lower bitrate (1 Mbps) compared to the link between `V0` and `V1` (8 Mbps). Therefore, the segmentation process splits the SFC requested into 2 segments, named `Seg.1`, which encompasses VNFs `V0` and `V1`, and `Seg.2`, which contains only VNF `V2`. The segments are created based on the compute and network resources available in the subjacent zone, using the aggregated data as the data source. For this example, all compute zones can execute all required VNFs.

The process to find the suitable compute zone to execute each segment is found through playing multiple games. The game objective is to identify which child zone should execute the VNFs inside the segment (player). The first game is executed by `A1` (coordination zone), and the players are segments `Seg.1` and `Seg.2`. In this example, `Seg.1` will be forwarded to zone `A3` and `Seg.2` to zone `A4`. Then, zones `A3` and `A4` will execute, in parallel, games to determine whether each child zone should execute the VNFs of those segments. This cycle will continue until each segment reaches a compute zone. When a compute zone is selected, the VNFs of the segment will be placed inside the zone. All VNFs inside a segment are always allocated in the same compute zone. The same segment can be part of many games, as `Seg.2` was part of the games executed in `A1` (Game 1) and `A4` (Game 2).

As selecting each zone for each segment is executed in parallel, the manager zone must be informed of the process by the child zones. When a segment reaches a compute zone, this zone sends a message to the coordination zone informing that the VNFs of the segment will be executed in that zone. The distributed allocation will only be considered finished when all requested VNFs are assigned to one compute zone. After the execution of the games, zone `A1` (coordination zone) creates the network connectivity between all zones used to execute the VNFs.

The time complexity of Algorithm 1 is $\mathcal{O}(I \cdot S \cdot Z)$, where S is the number of segments, Z is the number of candidate zones, and I is the number of iterations required to reach equilibrium. Although I may vary depending on the scenario, in practice, it tends to be small due to the fast convergence properties of singleton congestion games.

4.4.1 Players strategy update

We modeled the SFC placement problem as an SGC, as described in Section 4.4. 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.

There are several well-known approaches for updating player strategies in SGC games, comprising reinforcement learning (RL), Fictitious play, and no-regret learning [51]. However, some approaches are not suitable for specific scenarios. For example, prior studies have shown that RL fails

to converge when agents lack full observability of the environment [52].

In our proposed solution, we adopt an approach based on the best response dynamics (BRD) method [53]. 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.

A strategy update based on BRD is well suited to our problem, given that each player needs to be aware of the strategies selected by the other players. These strategies directly influence the cost of executing a segment within a given aggregation zone, as resource competition and shared infrastructure affect performance. Additionally, BRD aligns with the distributed nature of the system, allowing each segment to make decisions based on local cost evaluations.

Figure 6 illustrates the activity diagram that depicts the execution of the strategy based on the BRD approach.

The algorithm starts when an aggregation zone AZ receives a request. If it is the first zone involved in the SFC placement process, the request is treated as an *SFC placement request*, indicating that AZ is the coordination zone. Otherwise, the request contains an SFC segment forwarded by another aggregation zone.

After receiving the request, the aggregation zone AZ verifies whether all the requested VNFs can be executed within its own compute zone, based on its aggregated data. If this is the case, the game terminates; otherwise, the SFC segmentation algorithm, described in Section 4.3, is executed. This algorithm produces a segmentation plan represented by the list $K = \{k_1, k_2, \ldots, k_n\}$, where each element k_i corresponds to a segment to be executed in one of the available zones.

Then, the current aggregation zone AZ starts the game using the list K as input. The aggregation zone AZ is respon-

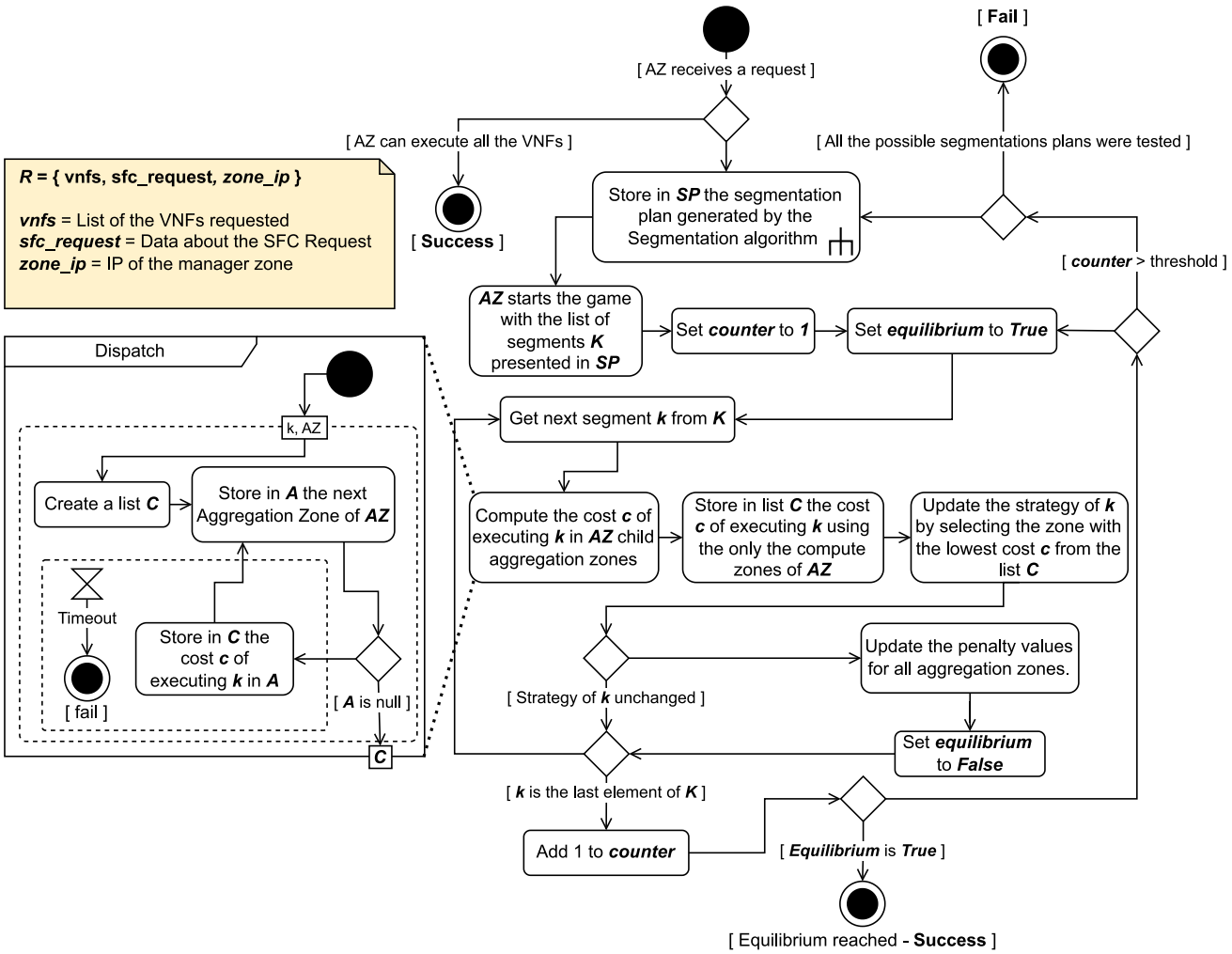


Fig. 6 Players strategy update activity diagram

sible for executing the game that determines the execution zone for each segment of the segmentation plan `k`. Each segment `k_i` can be executed either within the `AZ`'s own compute zones or in one of its child aggregation zones.

The game starts by initializing two control variables: the equilibrium flag and the counter. At the beginning of each major iteration, the algorithm resets the equilibrium flag by assigning `equilibrium` $\leftarrow$ `True`. This assumption will only hold if no segment changes its strategy during the current loop. If any segment finds a better execution zone, this flag will be changed to `False`, and the process will repeat. For each segment `k = K[i]`, the algorithm computes its execution cost `c` for all available zones.

The cost `c` of executing a segment in a given zone is defined by two factors: (i) the base VNF execution cost stored in the aggregated data and (ii) the level of congestion within the zone (penalty). Congestion arises from the strategies of players who select the same aggregation zones. As congestion in a zone increases, the execution cost grows, reflecting the impact of resource competition.

The penalty represents an additional cost that reflects the level of congestion within the zone. There are multiple strategies to apply penalties, such as linear, quadratic, and exponential. In linear approaches, the cost scales linearly with the number of assigned segments, while in quadratic approaches, the cost scales with the square of that number. Both approaches are computationally feasible for scenarios with up to a few hundred segments, depending on the available resources and the acceptable runtime overhead.

In critical environments, such as in some 5G use cases and edge-cloud IoT systems, applying an exponential penalty is particularly effective. This approach enables a rapid increase in the execution cost as a zone becomes more congested, thus discouraging workload concentration. Equations 10, 11, and 12 present how the cost `c` is computed.

$$c(k_i, z) = \left(\sum_{f \in \mathcal{F}(k_i)} c_{\text{vnf}}(f, z) \right) \cdot penalty_z \tag{10}$$

$$penalty_z = \alpha \cdot e^{\beta n_z} \tag{11}$$

$$n_z = \sum_{k \in \mathcal{K}(z)} |\mathcal{F}(k)| \tag{12}$$

where:

- $c(k_i, z)$ is the total cost of executing segment k_i in zone z. This value is obtained in the aggregated data stored at zone z. This value is only computed if all the VNFs of k_i could be executed in zone z.
- $\mathcal{F}(k_i)$ are the VNFs that compose segment k_i.
- $c_{\text{vnf}}(f, z)$ is the cost of execute a VNF f in the zone z.
- $\sum_{f \in \mathcal{F}(k_i)} c_{\text{vnf}}(f, z)$ is the base cost, computed as the sum of the costs of all VNFs that compose segment k_i when executed in zone z.
- $penalty_z$ is a multiplicative factor applied to the base cost to reflect the congestion level in zone z, defined by the number of VNFs currently deployed in the zone. Our approach considers the number of VNFs rather than their individual resource consumption, as resource-intensive VNFs naturally result in higher base costs.
- α is the scaling factor that adjusts the impact of congestion.
- β is the growth rate parameter controlling how rapidly the penalty increases.
- n_z is the total number of VNFs associated with all segments currently allocated to zone z;
- $\mathcal{K}(z)$ are segments already assigned to zone z.
- $|\mathcal{F}(k)|$ is the number of VNFs that compose segment k.

The cost function $c(k_i, z)$ is evaluated for `AZ` compute zones and in each `AZ` child aggregation zone. When the cost `c` is computed within the aggregation zone `AZ`, it considers only the capabilities of its child compute zones. In contrast, when computed by a child aggregation zone of `AZ`, the cost is based on the aggregated data obtained from its subordinate zones.

After evaluating the costs, the segment k_i selects the zone with the lowest execution cost. This is equivalent to the segment performing a best response action in a one-player optimization context. The updated strategy replaces the previous allocation for k_i if the cost of the previously selected zone is higher. If the newly chosen strategy for k_i is different from its previous one, then the system is no longer in equilibrium. To reflect this, the equilibrium flag is set to false again.

When a segment changes its assigned zone, the system updates the penalty values for all aggregation zones. This ensures that the penalty used in the cost computation accurately reflects the current level of congestion in the environment. These values must be updated for all zones, not only for those currently assigned to segments, since a zone may become unassigned and thus its congestion state must also be redefined.

After processing a segment, the game is performed using the next segment. If the end of the list is reached, the counter is incremented as well `counter` $\leftarrow$ `counter` $+ 1$. This mechanism tracks how many full iterations were executed.

The loop continues until one of two conditions is met: (i) no strategy changes are detected, indicating that an equilibrium has been reached and a stable solution has been found, or (ii) the maximum number of iterations is reached, defined as `counter > threshold`, which serves as a safeguard

Table 4 Simulation parameter settings

Description	Values
Number of domains	64
Compute nodes in domain	[5,10]
CPU capacity	[1,5] GHz
Memory capacity	[1,5] GB
Inter-domain links capacity	1000 Mbps
Intra-domain links capacity	2000 Mbps
Link delay	[2, 5] ms
CPU cost	[0.001, 0.005] $/s
Memory cost	[0.001, 0.004] $/GBs
CPU demand of VNF	[1,4] GHz
Memory demand of VNF	[1,5] GB
Traffic rate requirement	[100, 500] kbps
Traffic routing cost	[0.02, 0.05] $/Mb
Maximum tolerated delay	[0, 50] ms

against non-convergent behavior. In the second case, the algorithm performs another segmentation, and the entire process is executed again, starting from the generation of a new segmentation plan. The algorithm is considered to have failed only when all possible segmentation plans have been tested, and none of the corresponding games has reached equilibrium.

Although our algorithm currently employs an exponential penalty, it can be easily adapted to incorporate alternative penalty strategies by modifying the corresponding equation. Regarding the computation of the cost c, this process can be parallelized across all available aggregation zones, thereby reducing the overall execution time.

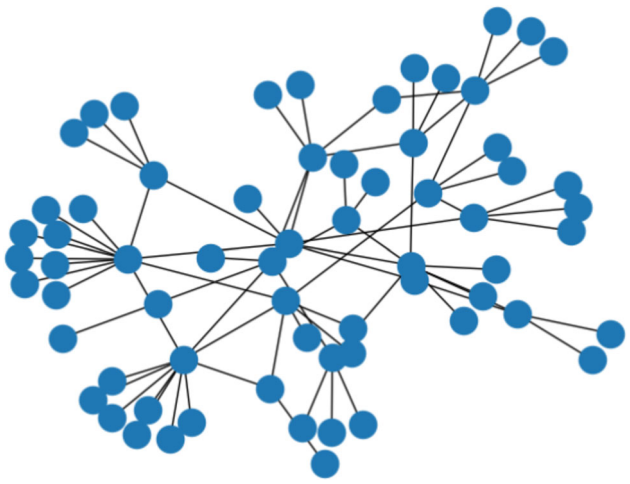


Fig. 7 Internode topology

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 at https://github.com/anselmobattisti/speed. The results presented in Section 5.1 were partially presented in our previous work [54].

Table 4 shows the simulation parameters, which were defined based on [42]. Figure 7 depicts the Internodes backbone topology, from the Internet Topology Zoo [55], used to organize the distributed environment. This topology has 64 nodes. Figure 8 depicts a fragment of how the zones were organized. 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 VNF types in the SFC represent network services like firewalls, intrusion detection, and cache. The CPU and memory requirements for each VNF type were obtained from the respective website. The simulations were executed ten times for each scenario, and the results are reported as mean values. We computed the 95% confidence interval assuming a normal distribution, which is reasonable given the sample size and the observed variance across repetitions.

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

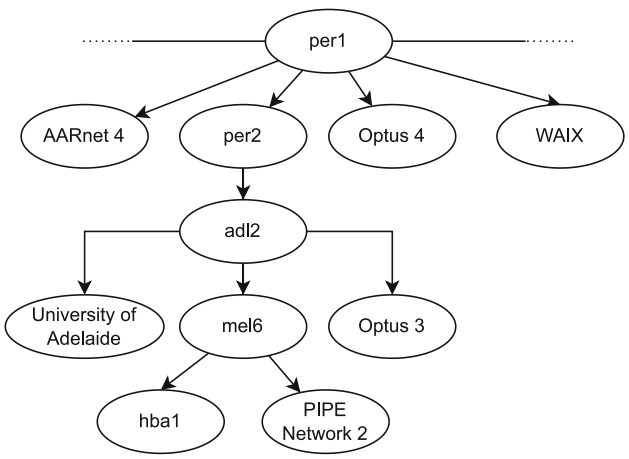
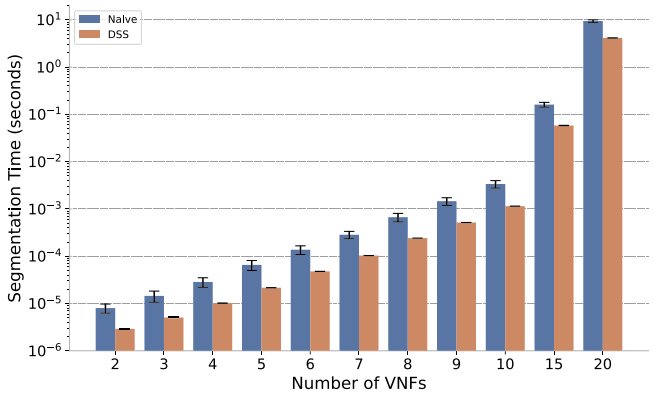
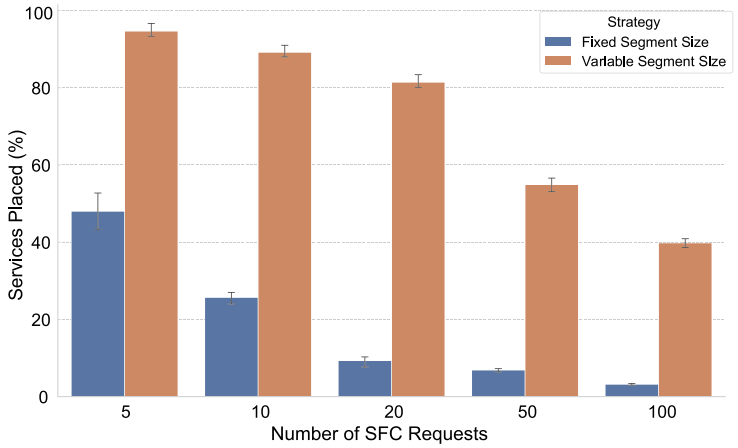


Fig. 8 Topology example

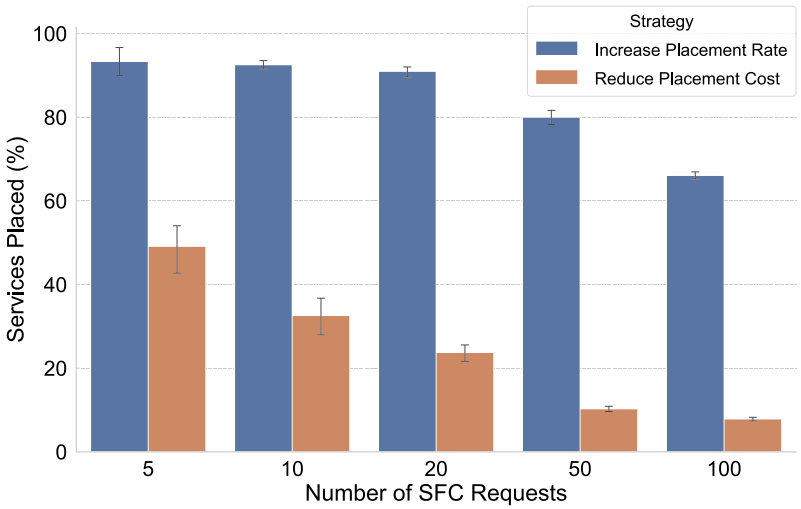
Fig. 9 Evaluation of SFC segmentation regarding time and success rate



(a) Segmentation Time by SFC Size



(b) SFC Placement Success Rates



(c) Impact of Cost Minimization on SFC Placement Success Rates

plans, and (ii) the *Service Placed*, that measures the percentage of SFCs placed.

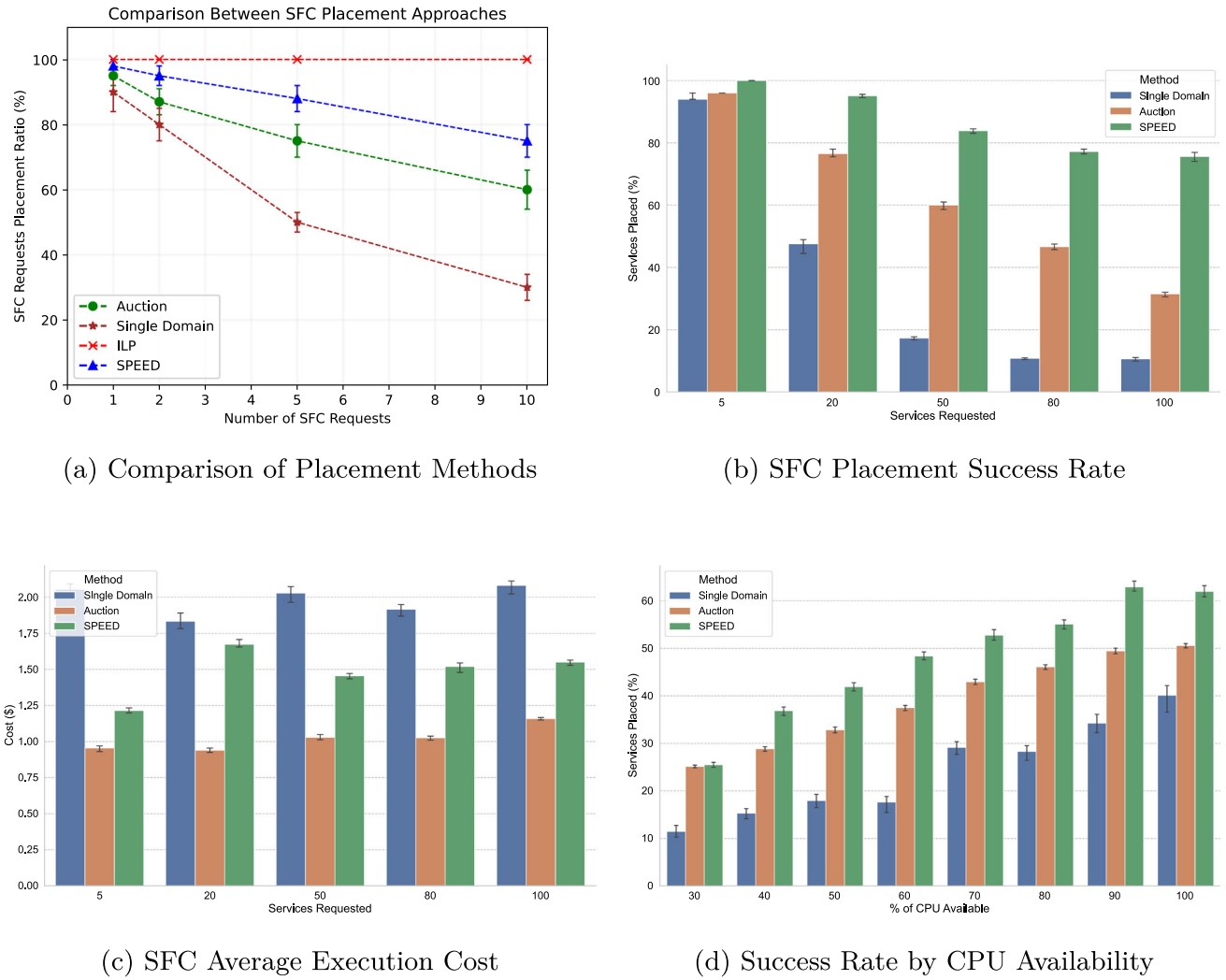
Figure 9a shows the average time required to generate the segmentation plan as the length of the SFC increases. The x-axis represents the number of VNFs, while the y-axis indicates the average generation time on a logarithmic scale. We compare our approach with a naive method that exhaustively generates 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 negatively impacting the total SFC placement time.

Figure 9b 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.

Figure 9c presents the execution of the SFC placement based on a greedy approach using two segmentation strategies: (i) tries to reduce the placement cost by selecting the nodes with lower execution cost and (ii) tries to increase the number of placed SFCs. The x-axis represents the number of SFC requests [5, 10, 20, 50, 100], and the y-axis shows the SFCs successfully placed. The results demonstrate that the cost reduction strategy tends to reduce the SFC placement success rate, particularly as the number of requests increases.

5.2 Distributed SFC placement

This section presents the results of our performance evaluation for SPEED. Our approach solves the SFC placement



(a) Comparison of Placement Methods

(b) SFC Placement Success Rate

(c) SFC Average Execution Cost

(d) Success Rate by CPU Availability

Fig. 10 Evaluation of SFC placement and execution cost under different conditions

problem in the edge-cloud continuum through data aggregation and game theory. This unique characteristic imposes challenges for a fair comparison with the solutions already proposed for the SFC placement problem. Therefore, we compare our work against the following approaches.

- **Single domain**: This is a decentralized approach that selects the resources to execute the SFC in the same domain where the request arrives. This can be considered a naive approach.
- **Auction**: This is a distributed approach that determines the execution domain for the SFC using an auction-based strategy. We include this method in our comparison because auction mechanisms are recognized in the literature as effective solutions for resource allocation in distributed environments [43, 47, 56]. Their key advantages are scalability, decentralization, and support for economic incentive models, thus making the auction approach a strong baseline for evaluating our approach.
- **Integer linear programming (ILP)**: This is a centralized method that generates the best response, based on the system model defined in Section 3.

We compare the approaches using 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.

Figure 10a 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 ten 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 the next experiments.

In Fig. 10b, 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. Figure 10c compares the average cost to execute each requested SFC.

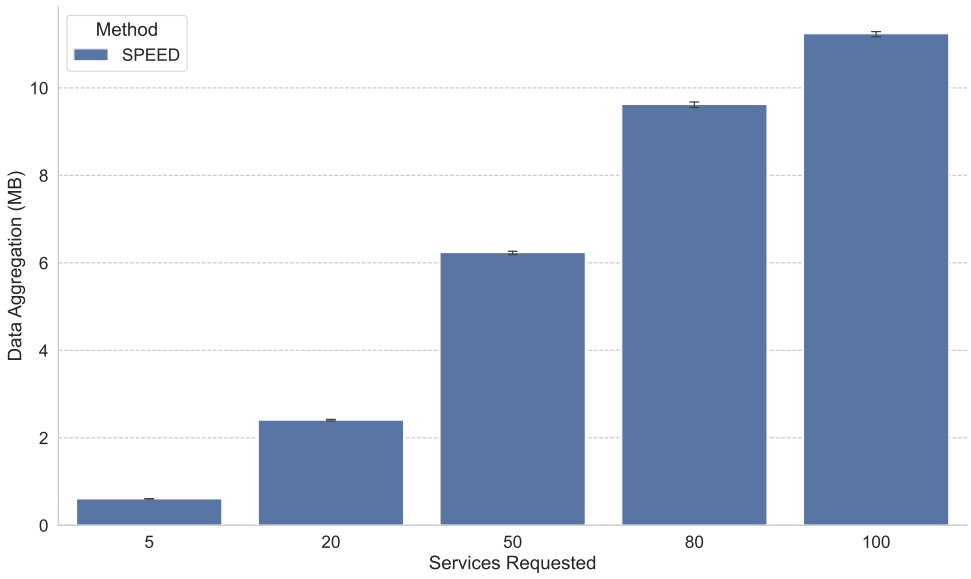


Fig. 11 Metadata consolidation size

The results show that even with a higher success placement rate, the average cost to execute each SFC was similar to the other approaches. Therefore, our proposed approach, despite increasing the number of requests placed, maintains similar costs in the execution of each SFC.

By adopting our approach, the infrastructure becomes more efficient due to optimized resource usage. As a result, service providers can increase their revenue. This gain is achieved by serving more clients or handling higher volumes of traffic at similar prices, while maintaining the SLA.

In a 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. Figure 10d 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.

Figure 11 presents the size of the metadata shared between the aggregation zones. The shared metadata grows proportionally to the number of requested services. We adopt a reactive approach in terms of metadata sharing; thus, we request the info about the children zones only when the zone is selected to participate in a game to execute an SFC segment. Regarding placement time, the SPEED approach took an average of 27 ms with a std of 2.1 ms to find the placement plans. Most of the time, there is a network delay caused by variations in topology and delays between zones.

6 Conclusion

In this work, we proposed a new distributed approach to solving SFC placement problem (SFCPP) called SPEED, which incorporates several novel characteristics. First, we adopt a recursive segmentation process that significantly advances this area's state-of-the-art. Second, we model the SFCPP using game theory, which provides a new way to understand and solve the SFC placement problem.

The experimental results indicated that, compared to other techniques, the SPEED approach increases the number of successful services placed by 30%. The results also show that SPEED, in an environment with limited resources, is more effective than the other approaches. Despite the increase in the number of requests placed, our proposed method ensures that the execution costs of each service remain consistent.

A major advantage of SPEED lies in minimizing inter-domain communication related to network topology and computational resource availability. This reduces communication overhead and preserves privacy in multi-domain environments. Additionally, the use of game theory improves the SFC placement success rate, enhancing the overall efficiency of the system.

This combination of distributed segmentation and congestion game theory in a distributed, privacy-preserving approach allows the approach to scale across complex multi-domain infrastructures. Its ability to operate without centralized knowledge makes it suitable for real-world deployments. These characteristics represent a meaningful contribution to the SFCPP literature, particularly in edge-cloud environments.

As future work, we will implement the proposed approach as a component within platforms such as Open Source MANO (OSM) or the Edge Multi-Cluster Orchestrator (EMCO). This implementation will allow us to perform evaluations of our approach in real-world scenarios. In addition, we intend to perform a comparative analysis between our approach and the latest state-of-the-art solutions, varying the different parameters that determine the requirements of the VNFs and also of the edge-cloud environment topology [57] with more complex network topologies.

Author contribution A.L.E.B. wrote the main manuscript text, implemented the proposed algorithm, and executed the experiments. F.C.D. reviewed the main manuscript text and analyzed the obtained results. D.M.S. reviewed the main manuscript text and validated the proposed algorithms. All authors approved the final manuscript and were accountable for its aspects, ensuring accuracy and integrity.

Funding The Article Processing Charge (APC) for the publication of this research was funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) (ROR identifier: 00x0ma614). This work was supported by CAPES, CNPq, MCTI, FAPERJ, and INCT-ICoNIoT. Flavia C. Delicato and Débora Muchaluat-Saade are CNPq and FAPERJ Fellows. The Article Processing Charge for the publication of this research was funded by the Coordination for the Improvement of Higher Education Personnel - CAPES (ROR identifier: 00x0ma614).

Data availability No datasets were generated or analysed during the current study.

Code availability The source code of the simulator and the implementation of the proposed approach are available at https://github.com/anselmobattisti/speed

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tutor 19(4):2322–2358. https://doi.org/10.1109/COMST.2017.2745201
2. Pan J, McElhannon J (2018) Future edge cloud and edge computing for Internet of Things applications. IEEE Internet Things J 5(1):439–449. https://doi.org/10.1109/JIOT.2017.2767608
3. Yi B, Wang X, Li K, k Das S, Huang M (2018) A comprehensive survey of network function virtualization. Comput Netw 133:212–262. https://doi.org/10.1016/j.comnet.2018.01.021
4. Mijumbi R, Serrat J, Gorricho JL, Bouten N, De Turck F, Boutaba R (2016) Network function virtualization: state-of-the-art and research challenges. IEEE Commun Surv Tutor 18(1):236–262. https://doi.org/10.1109/COMST.2015.2477041
5. Santos GL, Bezerra DdF, Rocha ÉdS, Ferreira L, Moreira ALC, Gonçalves GE et al (2022) Service function chain placement in distributed scenarios: a systematic review. J Netw Syst Manag 30(1):4
6. Laghrissi A, Taleb T (2019) A survey on the placement of virtual resources and virtual network functions. IEEE Commun Surv Tutor 21(2):1409–1434
7. Mostafavi S, Hakami V (2021) Quality of service provisioning in network function virtualization: a survey. Computing 103(5):917–991
8. Kaur K, Mangat V, Kumar K (2020) A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture. Comput Sci Rev 38:100298
9. Bhamare D, Jain R, Samaka M, Erbad A (2016) A survey on service function chaining. J Netw Comput Appl 75:138–155. https://doi.org/10.1016/j.jnca.2016.09.001
10. Kapassa E, Touloupou M, Kyriazis D (2018) SLAs in 5G: a complete framework facilitating VNF and NS tailored SLAs management. In: 2018 32nd International conference on advanced information networking and applications workshops (WAINA). vol 1, pp 469–474
11. Wang S, Cao H, Yang L (2020) A survey of service function chains orchestration in data center networks. In: 2020 IEEE Globecom workshops (GC Wkshps, pp 1–6 . 2020 IEEE Globecom Workshops (GC Wkshps. Taipei, Taiwan: IEEE
12. Kuo TW, Liou BH, Lin KCJ, Tsai MJ (2018) Deploying chains of virtual network functions: on the relation between link and server usage. IEEE/ACM Trans Networking 26(4):1562–1576. https://doi.org/10.1109/TNET.2018.2842798
13. Morabito R, Cozzolino V, Ding AY, Beijar N, Ott J (2018) Consolidate IoT edge computing with lightweight virtualization. IEEE Network 32(1):102–111. https://doi.org/10.1109/MNET.2018.1700175
14. Battisti ALÉ, Macedo ELC, Josué MIP, Barbalho H, Delicato FC, Muchaluat-Saade DC et al (2022) A novel strategy for VNF placement in edge computing environments. Future Internet 14(12):361
15. Toumi N, Bernier O, Meddour DE, Ksentini A (2021) On cross-domain service function chain orchestration: an architectural framework. Comput Netw 187. https://doi.org/10.1016/j.comnet.2021.107806
16. Ruiz L, Barroso RJD, De Miguel I, Merayo N, Aguado JC, De La Rosa R et al (2020) Genetic algorithm for holistic VNF-mapping and virtual topology design. IEEE Access 8:55893–55904. https://doi.org/10.1109/ACCESS.2020.2982018
17. Kim Si, Sung Kim H (2020) A VNF placement method considering load and hop count in NFV environment. In: 2020 International conference on information networking (ICOIN), pp 707–712. ICOIN. IEEE
18. Emu M, Yan P, Choudhury S (2020) Latency-aware VNF deployment at edge devices for IoT services: an artificial neural network-based approach. In: Proceedings of the 2020 IEEE international conference on communications workshops (ICC Workshops), pp 1–6. Proceedings of the 2020 IEEE international conference on communications workshops (ICC Workshops). IEEE
19. Sharif Z, Jasser MB, Yau KLA, Amphawan A (2025) Advancements and challenges in latency-optimized joint SFC placement and routing: a comprehensive review and future perspectives. Int J Syst Assur Eng Manag 16(3):1072–1105
20. Rasmusen E (2006) Games and information: an introduction to game theory. Wiley
21. Owen G (2013) Game theory. Emerald Group Publishing Limited, Bingley, UK
22. Rosenthal RW (1973) A class of games possessing pure-strategy Nash equilibria. Internat J Game Theory 2(1):65–67
23. Maskin E (1999) Nash equilibrium and welfare optimality. Rev Econ Stud 66:23–38. Reprinted in J.J. Laffont (ed), The Principal Agent Model: The Economic Theory of Incentives, London: Edward Elgar, 2003
24. Gairing M, Schoppmann F (2007) Total latency in singleton congestion games. In: Deng X, Graham FC (eds) Internet and Network Economics. Berlin, Heidelberg, Springer, Berlin Heidelberg, pp 381–387
25. Sbabou S, Smaoui H, Ziad A (2012) Equilibrium analysis in singleton congestion games. Centre d'Économie et de Management de l'Océan Indien (CEMOI), Université de La Réunion. 2013-04. Accessed 28 July 2025
26. Bilò V, Vinci C (2017) On the impact of singleton strategies in congestion games. In: 25th Annual european symposium on algorithms (ESA 2017), pp 17:1–17:14. vol 87 of Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik
27. Garrich M, Romero-Gázquez ea (2020) IT and multi-layer online resource allocation and offline planning in metropolitan networks. J Light Technol 38(12):3190–3199. https://doi.org/10.1109/JLT.2020.2990066
28. Li D, Hong P, Xue K, Pei J (2019) Virtual network function placement and resource optimization in NFV and edge computing enabled networks. Comput Netw 152:12–24
29. Nguyen DT, Pham C, Nguyen KK, Cheriet M (2020) Placement and chaining for run-time IoT service deployment in edge-cloud. IEEE Trans Netw Serv Manage 17(1):459–472. https://doi.org/10.1109/TNSM.2019.2948137
30. Reyhanian N, Farmanbar H, Mohajer S, Luo ZQ (2020) Joint resource allocation and routing for service function chaining with in-subnetwork processing. In: ICASSP 2020 - 2020 IEEE International conference on acoustics, speech and signal processing (ICASSP), pp 4990–4994
31. Kiran N, Liu X, Wang S, Yin C (2020) VNF Placement and Resource Allocation in SDN/NFVEnabled MEC Networks. In: 2020 IEEE Wireless communications and networking conference workshops (WCNCW), pp 1–6. WCNCW. IEEE
32. Pei J, Hong P, Pan M, Liu J, Zhou J (2020) Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks. IEEE J Sel Areas Commun 38(2):263–278
33. Bunyakitanon M, Vasilakos X, Nejabati R, Simeonidou D (2020) End-to-End Performance-Based Autonomous VNF Placement With Adopted Reinforcement Learning. IEEE Trans Cognit Commun Netw 6(2):534–547. https://doi.org/10.1109/TCCN.2020.2988486

34. Pham C, Tran NH, Ren S, Saad W, Hong CS (2020) Traffic-Aware and Energy-Efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach. IEEE Trans Serv Comput 13(1):172–185
35. Li S, Zhang S, Chen L, Chen H, Liu X, Lin S (2020) An Attention Based Deep Reinforcement Learning Method for Virtual Network Function Placement. In: 2020 IEEE 6th International conference on computer and communications (ICCC), pp 1005–1009
36. Niu M, Cheng B, Chen JL (2020) GPSO: A Graph-based Heuristic Algorithm for Service Function Chain Placement in Data Center Networks. In: 2020 IEEE International conference on services computing (SCC), pp 256–263. SCC. IEEE
37. Alahmad Y, Agarwal A, Daradkeh T (2020) Cost and Availability-Aware VNF Selection and Placement for Network Services in NFV. In: 2020 International symposium on networks, computers and communications (ISNCC), pp 1–6
38. Gao T, Li X, Wu Y, Zou W, Huang S, Tornatore M et al (2020) Cost-Efficient VNF Placement and Scheduling in Public Cloud Networks. IEEE Trans Commun 68(8):4946–4959. https://doi.org/10.1109/TCOMM.2020.2992504
39. Mohamad A, Hassanein HS (2019) On Demonstrating the Gain of SFC Placement with VNF Sharing at the Edge. In: 2019 IEEE Global communications conference, pp 1–6. GLOBECOM
40. Vaquero LM, Cuadrado F, Elkhatib Y, Bernal-Bernabe J, Srirama SN, Zhani MF (2019) Research challenges in nextgen service orchestration. Futur Gener Comput Syst 90:20–38. arXiv:1806.00764
41. Huff A, Venâncio G, Jr ED (2019) Multi-SFC: Orquestração de SFCs Distribuídas sobre Múltiplas Nuvens em Múltiplos Domínios com Múltiplas Plataformas NFV. In: Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, pp 777–790. Porto Alegre, RS, Brasil: SBC
42. Chen C, Nagel L, Cui L, Tso FP (2022) Distributed federated service chaining: A scalable and cost-aware approach for multi-domain networks. Comput Netw 212:109044. https://doi.org/10.1016/j.comnet.2022.109044
43. Liu Y, Zhang H, Chang D, Hu H (2020) GDM: A General Distributed Method for Cross-Domain Service Function Chain Embedding. IEEE Trans Netw Serv Manage 17(3):1446–1459. https://doi.org/10.1109/TNSM.2020.2993364
44. Zavlanos M, Spesivtsev L, Pappas GJ (2008) A distributed auction algorithm for the assignment problem. In: Proceedings of the IEEE conference on decision and control (CDC), pp 1212–1217. Proceedings of the IEEE Conference on Decision and Control (CDC)
45. Sun G, Li Y, Liao D, Chang V (2018) Service Function Chain Orchestration Across Multiple Domains: A Full Mesh Aggregation Approach. IEEE Trans Netw Serv Manage 15(3):1175–1191. https://doi.org/10.1109/TNSM.2018.2861717
46. Lee WC (1995) Topology aggregation for hierarchical routing in ATM networks. Comput Commun Rev 25(2):82–92
47. Avasalcai C, Tsigkanos C, Dustdar S (2019) Decentralized Resource Auctioning for Latency-Sensitive Edge Computing. In: 2019 IEEE International conference on edge computing (EDGE), pp 72–76
48. Soualah O, Mechtri M, Ghribi C, Zeghlache D (2017) A link failure recovery algorithm for Virtual Network Function chaining. In: 2017 IFIP/IEEE Symposium on integrated network and service management (IM), pp 213–221
49. Xu Q, Gao D, Li T, Zhang H (2018) Low Latency Security Function Chain Embedding Across Multiple Domains. IEEE Access 6:14474–14484. https://doi.org/10.1109/ACCESS.2018.2791963
50. Stanley RP (2011) Enumerative Combinatorics: vol 1, 2nd edn. Cambridge University Press, USA
51. Fudenberg D, Levine DK (1998) The theory of learning in games, vol 2. MIT press, Cambridge, MA
52. Iwase T, Tadokoro Y, Fukuda D (2017) Self-Fulfilling Signal of an Endogenous State in Network Congestion Games. Netw Spat Econ 17(3):889–909. https://doi.org/10.1007/s11067-017-9351-4
53. Swenson B, Murray R, Kar S (2018) On Best-Response Dynamics in Potential Games. SIAM J Control Optim 56(4):2734–2767. https://doi.org/10.1137/17M1139461
54. Battisti ALE, Delicato FC, Muchaluat-Saade DC (2024) A Novel Method for SFC Segmentation in Edge-Cloud Environments. In: 2024 IEEE 13th International conference on cloud networking (CloudNet), pp 1–8. CloudNet. IEEE
55. Knight S, Nguyen HX, Falkner N, Bowden R, Roughan M (2011) The Internet Topology Zoo. IEEE J Sel Areas Commun 29(9):1765–1775. https://doi.org/10.1109/JSAC.2011.111002
56. Macedo ELC, Battisti ALE, Vieira JL, Noce J, Pires PF, Muchaluat-Saade DC et al (2023) Distributed Auction-Based SFC Placement in a Multi-domain 5G Environment. SN Comput Sci 5(1):48. https://doi.org/10.1007/s42979-023-02291-1
57. Albert R, Barabási AL (2002) Statistical mechanics of complex networks. Rev Mod Phys 74(1):47