UNIVERSIDADE FEDERAL FLUMINENSE

ANSELMO LUIZ ÉDEN BATTISTI

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

NITERÓI

2025

ANSELMO LUIZ ÉDEN BATTISTI

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

Thesis presented to the Programa de Pós-Graduação em Computação of the Universidade Federal Fluminense as a partial requirement to obtain the Doctor of Science degree in Computer Science. Field: Computer Science

Advisor:
FLÁVIA COIMBRA DELICATO

Co-Advisor:
DÉBORA CHRISTINA MUCHALUAT SAADE

NITERÓI

2025

ANSELMO LUIZ ÉDEN BATTISTI

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

Thesis presented to the Programa de Pós-Graduação em Computação of the Universidade Federal Fluminense as a partial requirement to obtain the Doctor of Science degree in Computer Science. Field: Computer Science

Approved in September 2025.

EXAMINATION BOARD

Prof[a]. FLÁVIA COIMBRA DELICATO - Advisor, UFF

Prof[a]. DÉBORA CHRISTINA MUCHALUAT SAADE - Co-Advisor, UFF

Prof. ANTONIO AUGUSTO DE ARAGÃO ROCHA, UFF

Prof. PAULO DE FIGUEIREDO PIRES, UFF

Prof[a]. THAIS VASCONCELOS BATISTA, UFRN

Prof. EDUARDO COELHO CERQUEIRA, UFPA

Niterói

2025

Acknowledgements

I want to thank God for giving me the strength to achieve one of the most aspired goals of my life. I also thank Him for blessing me with a wonderful son, Oliver, who was born in 2023, during the course of this PhD, and brought even more meaning to this journey. I carried Oliver in my heart and mind every step of this journey.

I am deeply grateful for the unconditional support of my beloved parents, Fátima and Janir, my loving partner Marielle, my sister Monalisa, my brother Abraão, my favorite niece Bianca, my brother-in-law André, and my sister-in-law Bruna.

I extend my sincere gratitude to my advisors, Professor Débora and Professor Flávia, for their dedication, guidance, and the patience they had with me throughout this work.

I am also thankful to my friend Fernando Hallberg for his support in this and many other projects, and to my friend Evandro for his valuable collaboration in several publications.

I would like to express my appreciation to the MídiaCom Lab for providing the infrastructure necessary to carry out the experiments in this thesis. A special thanks to Marister, who had our backs on multiple occasions. Last but not least, I thank everyone at the UFF Computing Institute for welcoming me so warmly.

Resumo

Recentemente, a virtualização dos recursos de computação e rede sobre a infraestrutura física ganhou força. Este paradigma foi adotado primeiramente nas funções do núcleo da rede, criando o conceito de *Network Function Virtualization*. O processo de seleção dos recursos utilizados para executar um conjunto de VNFs é chamado de placement. No entanto, virtualizar apenas VNFs individualmente pode ser insuficiente para atender aos requisitos do usuário. Assim, foi criada uma nova abordagem chamada *Service Function Chain*, uma cadeia ordenada de VNFs normalmente associada a um SLA (*Service Level Agreement*). Com o avanço do paradigma de computação Edge-Cloud, as SFCs puderam ser executadas em vários nós gerenciados por provedores de serviços distintos. Neste cenário, um mecanismo de *placement* distribuído deve ser adotado para alocar as SFCs sem conhecimento completo da infraestrutura. Esta tese propõe uma nova abordagem, denominada SPEED, que endereça o problema de *placement* de SFCs em um ambiente multidomínio de forma distribuída. A solução engloba os componentes arquiteturais, um novo modelo de mapeamento entre o Problema de *Placement* de SFCs e a abordagem de Teoria dos Jogos, além de uma nova estratégia de agregação de dados. Os resultados mostram que a abordagem SPEED é viável e, comparada com outras abordagens, tem um ganho de 20% no número de requisições alocadas.

Palavras-chave: SFC, *Placement* de SFCs, *Placement* de SFCs Distribuído.

Abstract

In the last decades, there has been a trend to virtualize computing and networking resources over the physical infrastructure. This paradigm was first adopted in the network core functions, thus creating the Network Function Virtualization concept. The process of selecting the resource to execute a set of Virtual Network Functions (VNF) is named placement. However, virtualizing only VNFs individually is often not enough to meet the user's requirements. Hence, a new approach called Service Function Chain was created, an ordered chain of VNFs typically associated with one SLA (Service Level Agreement). With the advance of the Edge-Cloud computing paradigm, SFCs can be executed in multiple nodes managed by independent service providers. In this scenario, a distributed placement mechanism should be created to allocate SFCs without complete knowledge of the infrastructure where the corresponding VNFs will be executed. This thesis proposes a new approach, named SPEED, to solve the SFC Placement Problem over a multi-domain environment in a distributed fashion. The solution encompasses the architectural components, a new mapping model between the SFC Placement Problem and the Game-Theory approach, besides a new metadata aggregation strategy. The results show that SPEED is feasible and, compared with other approaches, has a gain of 20% in placed requests success.

Keywords: SFC, SFC Placement, Distributed SFC Placement.

List of Figures

List of Tables

List of Acronyms

API Application Programming Interface

BSS Business Suport System

CAPEX Capital Expenditure

DSFCPP Distributed SFC Placement Problem

ETSI European Telecommunications Standards Institute

GT Game Theory

IoT Internet of Things

KVM Kernel-based Virtual Machine

MANO Management and Orchestration

NAT Network Address Translation

NE Nash Equilibrium

NFV Network Function Virtualization

NFVI NFV Infrastructure

NFVO NFV Orchestrator

O-RAN Open RAN

OPEX Operational Expenditure

OSM Open Source MANO

OSS Operation Suport System

RAN Radio Access Network

SCG Singleton Congestion Game

SDN Software Defined Networking

SF Service Functions

SFC Service Function Chain

SFCPP SFC Placement Problem

SFCPPlan SFC Placement Plan

SFF Service Functions Forwarders

SFP Service Functions Path

SLA Service Level Agreement

SPEED SFC Placement in Edge-Cloud Continuum

VIM Virtualized Infrastructure Manager

VNF Virtual Network Function

VPN Virtual Private Network

Summary

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 was essential for the development of multiple computational environments like Cloud, Edge, and Internet of Things (IoT) (PAN et al., 2018). More recently, with the development of 5G (FOUKAS et al., 2017), virtualization has also been adapted to virtualize the telecommunication infrastructure.

This virtualization paradigm was first applied to the network core functions like 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 referred to as Virtual Network Function (VNF) (MIJUMBI et al., 2016). VNFs are software-based network services that provide a specific function such as routing, firewall, and load balancing (YI et al., 2018).

More recently, even high-level functions directly requested by users, such as video encoders and antivirus, are also virtualized as VNFs (SANTOS et al., 2022). 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 resource provisioning speed and efficiency, and facilitates the management and orchestration of network functions (YI et al., 2018; MAO et al., 2017; MIJUMBI et al., 2016; ALAM et al., 2020).

In traditional environments, the network functions were executed by physical devices, and the network team had to manually define the location of the physical network where the appliance hardware was installed. However, with the adoption of the NFV paradigm, this task could be automated. This process has introduced a new problem known as VNF placement, which determines the suitable computational node for the execution of a given VNF (LAGHRISSI et al., 2019). Various centralized and decentralized approaches have

been proposed to address the VNF Placement problem, including algorithms like the ones published in (RUIZ et al., 2020; LI, D. et al., 2019; REYHANIAN et al., 2020; NIU et al., 2020; BATTISTI; MACEDO et al., 2022; MACEDO et al., 2023).

Virtualizing only VNFs individually is often not enough to meet the user's service demands. Users commonly require complex services consisting of multiple interconnected VNFs (MOSTAFAVI et al., 2021). To address this, the concept of Service Function Chain (SFC) was introduced (KAUR et al., 2020; PATTARANANTAKUL et al., 2023).

An SFC is a chain of sorted VNFs, associated with a Service Level Agreement (SLA) (BHAMARE et al., 2016). The SLA is a set of rules that specify how a service must operate. These rules are derived from various sources, including key performance indicators (KPIs), low-level infrastructure demands, and high-level user requirements. For example, the SLA may specify the number of service replicas that must be maintained, the maximum permitted CPU load, or the maximum acceptable delay in response times. These rules are essential to ensure that the service meets the needs of its users and operates in a reliable and efficient manner (KAPASSA et al., 2018).

Therefore, with the demand to place multiple VNFs that compose the SFC, the SFC Placement Problem (SFCPP) has emerged (WANG et al., 2020). In this problem, new challenges must be addressed; beyond defining which compute node should be used to execute each VNF, the connectivity to the network should also be selected, allowing the flow through all VNFs of the SFC (MOHAMAD et al., 2020; ALAHMAD et al., 2020; BUNYAKITANON et al., 2020).

The final outcome of the SFCPP is a placement plan. The placement plan specifies which compute nodes will host each VNF and defines the network paths that ensure correct traffic flow through the SFC. This plan must consider resource constraints, latency and bandwidth requirements, and cost efficiency to optimize overall network performance (LI, C. et al., 2025).

There are numerous approaches to solve the SFCPP. Among the most prominent are Genetic Algorithms (RUIZ et al., 2020), Markov Chains (NGUYEN et al., 2020; GAO, T. et al., 2020), and Greedy strategies (MOHAMAD et al., 2019; GARRICH et al., 2020). Although Game Theory is not commonly applied to this type of problem, it offers a solid foundation for modeling the SFCPP as a Congestion Game (LI, J. et al., 2021), where resource costs depend on the number of players selecting them. This modeling enables a distributed strategy for SFC placement in multi-domain environments.

With the advance of Edge-Cloud computing, small to medium-sized compute nodes that aim to provide additional computing, storage, and networking resources to the applications deployed across Internet of Things (IoT) devices, edge, and cloud (MORABITO et al., 2018) became available. Thus, these compute nodes can cooperate with each other to meet the needs to execute an SFC. Therefore, the VNFs within an SFC can be deployed and executed across diverse and heterogeneous nodes, potentially under the management of different service providers (TOUMI et al., 2021).

In an Edge-Cloud scenario, most of the proposed solutions for finding the SFC placement plan adopt a centralized approach, however, this approach is not adequate, given the massive number of nodes in the environment, confidentiality problems, and time constraints to create the placement plan (SON et al., 2019). Adopting distributed solutions for the SFC Placement problem is negligibly studied and it can potentially reduce operational costs, decrease delay, optimize resource consumption, and increase revenue (SANTOS et al., 2022). Thus, distributed mechanisms should be provided to allocate SFCs without complete knowledge of the underneath infrastructure where their VNFs will be executed (CISNEROS et al., 2022).

1.1 Problem Statement and Goal

This section presents the problem statement regarding the SFC Placement Problem (SFCPP). SFCPP refers to the problem described as: **"given a requested SFC, a set of constraints and restrictions, and the meta-data about the infrastructure that provides compute and network resources, it finds the compute nodes to execute each VNF and identifies the network links that provide connectivity through the SFC"** (KAUR et al., 2020; BHAMARE et al., 2016; ALAM et al., 2020). This problem has been proven to be NP-Hard (KUO et al., 2018), which makes the development of effective heuristics crucial to finding near-optimal solutions.

Multiple strategies have been proposed to solve the SFCPP, some of them use a centralized approach (BUNYAKITANON et al., 2020; KIRAN et al., 2020; NGUYEN et al., 2020), others use a distributed approach (SUN et al., 2018; AVASALCAI et al., 2019; ZHANG, Z. et al., 2024). However, they tend to be inefficient when the number of entities involved increases. Thus, developing novel distributed solutions to reduce operational costs, reduce the total delay, and increase the SFC request execution rate in environments with multiple compute nodes, like multi-domain environments, is necessary.

To execute the requested services, computational and network resources must be allocated. These resources are owned by companies such as service providers or network operators, who wish to lease them (SHINDE et al., 2021). In this work, we define a domain as a set of computational and network resources managed by one company. A single company can arrange its resources into multiple domains. The resources in a domain can be organized into zones. Figure 1 depicts an example of a multi-domain environment scenario. The proposed approach was designed to address the SFCPP in a scenario where owners of multiple domains compete to lease their resources. We named this scenario an Edge-Cloud continuum multi-domain environment.

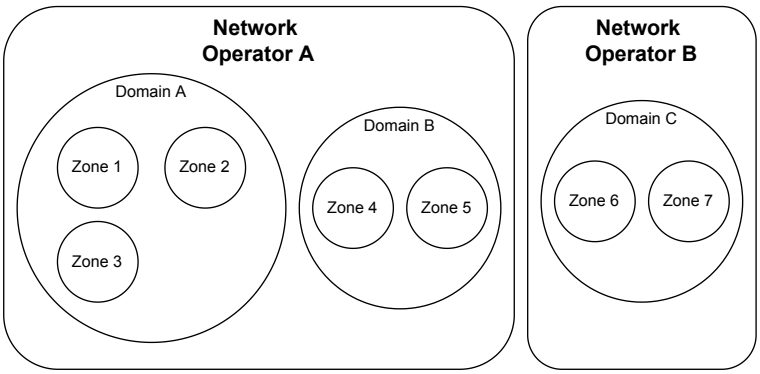


Figure 1: Example of a multi-domain environment scenario.

In scenarios where multiple entities make strategic decisions, for example, either leasing or not leasing a resource, based on the behavior of the other entities, the Game Theory (GT) can be applied. The GT is the study of strategic interactions between two or more players in situations where the result obtained by each player depends on the choices of other players (OWEN, 2013). The objective of the GT is to understand how rational players can make decisions in order to achieve an optimal outcome (ROY et al., 2010).

In this thesis, we propose the **SFC Placement in Edge-Cloud Continuum (SPEED)**. The proposed approach solves the SFCPP in a distributed manner, where the placement plan is created in parallel across the Edge-Cloud continuum. SPEED enables domains to collaboratively explore the solution space and to find a suitable placement plan, if one exists, for executing a requested SFC in a multi-domain environment.

The Edge-Cloud continuum is modeled as a hierarchical, tree-like topology comprising access, compute, and aggregation zones. Aggregation zones are responsible for maintaining metadata about all their subordinate zones. A metadata aggregation strategy, named as *data aggregation* in the remainder of this text, is employed to share information

about the VNF types that can be executed in each zone, thus supporting the distributed placement process.

The proposed algorithm decides which VNF (composing an SFC) should be executed in each zone. Such a decision is made by employing GT. In particular, our approach is based on multiple Congestion Games (ROSENTHAL, 1973). Congestion games are a class of strategic games used to model situations in which players compete against each other for the same resources. The goal of each player is to minimize the cost of using the resource, while the goal of the resource provider is to maximize revenue (OWEN, 2013; BILÒ et al., 2017).

Our proposed solution allows independent service providers to work together to execute the VNFs of a single SFC without fully revealing their network structure. SPEED also makes the environment more secure by keeping most of the network information private and helps improve the success rate of SFC requests. In the next section, we introduce the research questions that guide our study.

1.2 Research Questions

Based on the problem statement, the SFC Placement Problem (SFCPP) focuses on developing a suitable placement plan for an SFC request in a multi-domain environment. Building on this context, this thesis is guided by the following research questions:

- **Q1**: How can Game Theory be used to solve the Service Function Chain Placement Problem (SFCPP) in a multi-domain Edge-Cloud continuum?
  - *Method:* This question is addressed through the design and evaluation of a set of algorithms and an architecture that enable the use of Game Theory to solve the SFCPP. The proposed solution is validated by providing a detailed description of the algorithms and architecture, as well as by executing experiments in both simulated and real scenarios.
- **Q2:** How does the proposed approach contribute to reducing the monetary cost of executing SFCs in a multi-domain Edge-Cloud continuum?
  - *Method:* This question is answered by conducting cost-oriented performance evaluations using scenarios based on realistic infrastructure and traffic parameters.

- **Q3:** How does the proposed approach contribute to increasing the SFC placement success rate in a multi-domain Edge-Cloud continuum?
  - *Method:* This question is addressed through experiments measuring placement success rate under varying resource availability and network conditions.

1.3 Main Contributions

This thesis presents a set of original contributions developed in the context of the SFCPP in multi-domain environments. These contributions reflect the outcomes of the research and advances achieved throughout the study. Thus, the contributions of this thesis are:

1. A novel approach, named SFC Placement in Edge-Cloud Continuum (SPEED), for solving the SFC Placement Problem in multi-domain environments, adopting a distributed approach;
2. A novel architecture that enables the execution of SFCs across multiple domains;
3. A new system model that maps the SFC Placement Problem as a game, enabling placement resolution through Game Theory techniques;
4. A novel strategy for structuring metadata that enables the sharing of VNF metadata information across domains in distributed environments;
5. An original method for solving the SFC Segmentation Problem that can be used in multiple distributed SFC Placement algorithms.

1.4 Text Structure

The remainder of this document is organized as follows.

- Chapter 2 presents the theoretical background, introducing the main concepts required to understand the proposed approach depicted in this thesis.
- Chapter 3 describes and compares related studies on the SFC Placement Problem and existing architectures for multi-domain environments.
- Chapter 4 details our proposed approach, SFC Placement in Edge-Cloud Continuum (SPEED).
- Chapter 5 evaluates the proposed approach in a simulated environment.
- Chapter 6 presents the evaluation of the proposed approach in a real-world environment.
- Chapter 7 outlines the conclusions, discusses limitations, and suggests directions for future work.

2 Background

This chapter presents the key concepts and technologies relevant to this thesis. The NFV paradigm and NFV architecture are introduced in Sections 2.1 and 2.2, respectively. The SFC concept is discussed in Section 2.3, followed by an analysis of the SFC Placement Problem in Section 2.4, and its distributed variant in Section 2.5. Game Theory is addressed in Section 2.6, and finally, the SFC Segmentation Problem is presented in Section 2.7.

2.1 NFV Paradigm

The Network Function Virtualization (NFV) paradigm is a trend that has recently gained considerable momentum (COÊLHO et al., 2024). The increased complexity and cost regarding managing and orchestrating network functions have motivated the search for new approaches to address this issue (ZHANG, C. et al., 2019). Moreover, the NFV paradigm has already been adopted in a variety of scenarios such as IoT (OJO et al., 2016), 5G (TININI et al., 2020), and multimedia stream processing (VIOLA et al., 2023).

A key component of the NFV paradigm is the Virtual Network Function (VNF). VNF is a software-based network service that provides a specific network function such as routing, firewall, and load balancing (YI et al., 2018). Originally, the VNFs were virtualized versions of hardware appliances that are specifically designed to perform certain network functions. Nowadays, high-level functions are also virtualized in adopting this fashion (LAGHRISSI et al., 2019). VNFs can be deployed in heterogeneous environments like private, public, hybrid cloud, and edge data centers (XU, Z. et al., 2020).

Adopting new paradigms brings opportunities and challenges. Studies like (KAUR et al., 2020; ALAM et al., 2020; WANG et al., 2020; BHAMARE et al., 2016) pointed out multiple aspects that drive the NFV paradigm across multiple fields. Some of the key aspects presented are:

1. **Virtualization:** moving from the physical hardware that supports legacy network functions to a cloud-based virtualized one;
2. **Abstraction:** consider the network function as software-defined blocks executed in the environment;
3. **Automation:** automating required tasks of the network functions like placement, scaling, monitoring, and life cycle management;
4. **Programmability:** Enabling the adoption of open standards Application Programming Interface (API) to extend the network function management via external programmed solutions;
5. **Service Chaining:** is the ability to create a service with multiple virtualized network functions.

Besides the benefits presented, the adoption of the NFV paradigm also brings multiple challenges. Managing VNFs across multiple domains (KATSALIS et al., 2016) is one of them. When more than one domain is involved in the execution of VNFs, aspects regarding privacy, security, and business models increase the complexity of the management process.

Another aspect regarding the adoption of the NFV paradigm is the orchestration process (WANG et al., 2020). The orchestration involves the coordination and automation of diverse virtualized and physical components, such as VNFs, links, and other network functions (SOUSA et al., 2019). This process requires the ability to manage, configure, monitor, and control the entire NFV Infrastructure, discussed in detail in Section 2.2. Furthermore, orchestration must be able to ensure that these VNFs are deployed securely and efficiently (LI, X. et al., 2016) while guaranteeing the SLA scalability and flexibility.

2.2 NFV Platforms

Network Function Virtualization (NFV) is a concept that leverages virtualization to deploy network services more dynamically and cost-effectively (ZHANG, T. et al., 2021). NFV enables the decoupling of network functions from proprietary hardware, allowing them to be executed on commodity hardware (MIJUMBI et al., 2016). Software that provides these services is called Virtual Network Functions (VNFs).

To execute a VNF, an NVF Platform is required (KAUR et al., 2020; SCIANCALEPORE et al., 2016). There are numerous platforms available, such as Open Source

MANO (OSM)[1], Anuket[2] and ONAP[3]. Figure 2 presents the NFV Infrastructure (NFVI) and the Management and Orchestration (MANO), which are responsible for key tasks executed in NFV Platforms.

A promising approach to execute VNFs is to utilize containerization and orchestration tools. The Network Service Mesh (NSM) is an open-source solution designed to enhance Kubernetes networking, making it capable of supporting advanced scenarios required by NFV platforms. It allows network functions to connect through secure, policy-driven service meshes that are dynamically configured and not limited by traditional IP networking. This makes NSM a powerful tool for NFV environments that demand flexibility, scalability, and seamless integration across cloud and edge infrastructures (BITTAR et al., 2022).

The Operation Suport System (OSS) and Business Suport System (BSS) are systems that collaborate to offer services to users by providing interfaces for requesting and managing Virtual Network Function (VNF) and services (SOUSA et al., 2019). These systems are composed of multiple components, including client portals, APIs, billing, etc. They work together to ensure that the requested services are provided to the user.

The Management and Orchestration (MANO) component is responsible for a range of tasks related to the execution of VNFs and network services (MAMUSHIANE et al., 2019). These tasks include VNF and SFC Placement (YI et al., 2018), auto Scaling (CHEN, H.-L. et al., 2019), life cycle management, and network connectivity coordination, typically provided by an Software Defined Networking (SDN) (ALAM et al., 2020).

For decoupling the VNF from the hardware, the NFV Platform provides virtualization through the NFVI component (REHMAN et al., 2021). Popular hypervisors, such as XEN[4], Kernel-based Virtual Machine (KVM), and VMware[5], are used to virtualize resources consumed by VNFs. The VIM component provides the interfaces that enable the MANO component to request and monitor the VNF resources accordingly.

Typically, the MANO component has access only to local resources (domain resources). With the adoption of VNFs in new environments such as 5G and 6G, along with the emergence of new use cases (ETSI, 2019), the complexity of services provided is increasing. To address these scenarios, in a multi-domain environment, multiple NFV

[1]https://osm.etsi.org/
[2]https://anuket.io/
[3]https://www.onap.org/
[4]https://xenproject.org/
[5]https://www.vmware.com/br.html

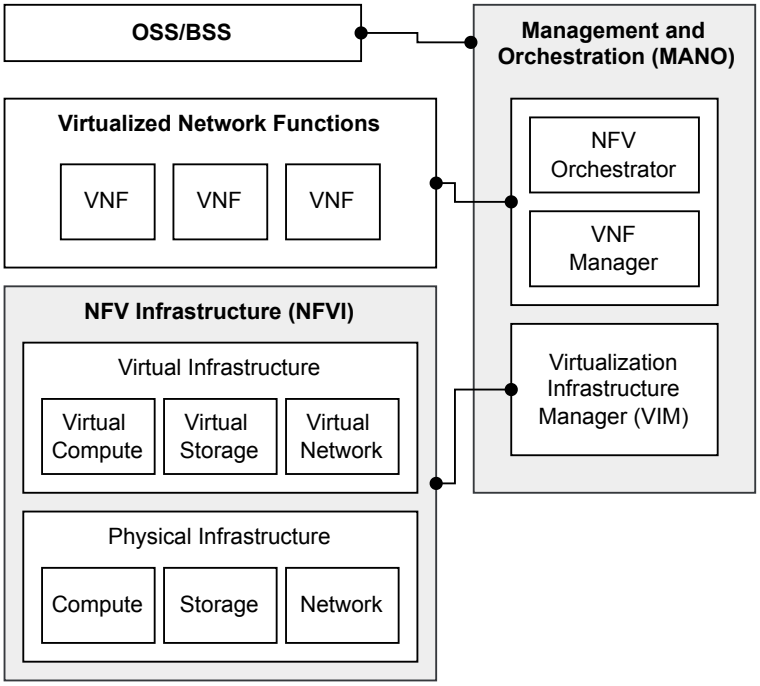


Figure 2: NFV Platforms main components.

platforms must coordinate their actions to execute the service requested by the user.

2.3 Service Function Chain

This section details the Service Function Chain (SFC) concept. An SFC can be defined as a network service composed of multiple VNFs (SANTOS et al., 2022). The data flow crosses the VNF chain via virtual links (BHAMARE et al., 2016) and has an SLA (TOUMI, NASSIMA AND BAGAA, MILOUD AND KSENTINI, ADLEN, 2021; YAGHOUBPOUR et al., 2022) that guides the orchestration component in the deployment phase and the life cycle manager component during the execution phase of the SFC. Figure 3 presents an overview of the elements that compose an SFC.

The structure of an SFC is defined by the IETF RFC 7665 (HALPERN et al., 2015), defining two basic aspects of the SFCs. The first one is the Service Functions (SF) that manages the ingress traffic, and the second one is the Service Functions Forwarders (SFF) that is responsible for steering the packet flow between the SFFs according to a pre-defined Service Functions Path (SFP) (TOUMI et al., 2020). However, the RFC defines only the scenario where the SFC is deployed on a single domain, and there are no references to multi-domain environment scenarios. Executing SFC over a multi-domain environment is much more complex than executing SFCs in a single domain, especially when the domain

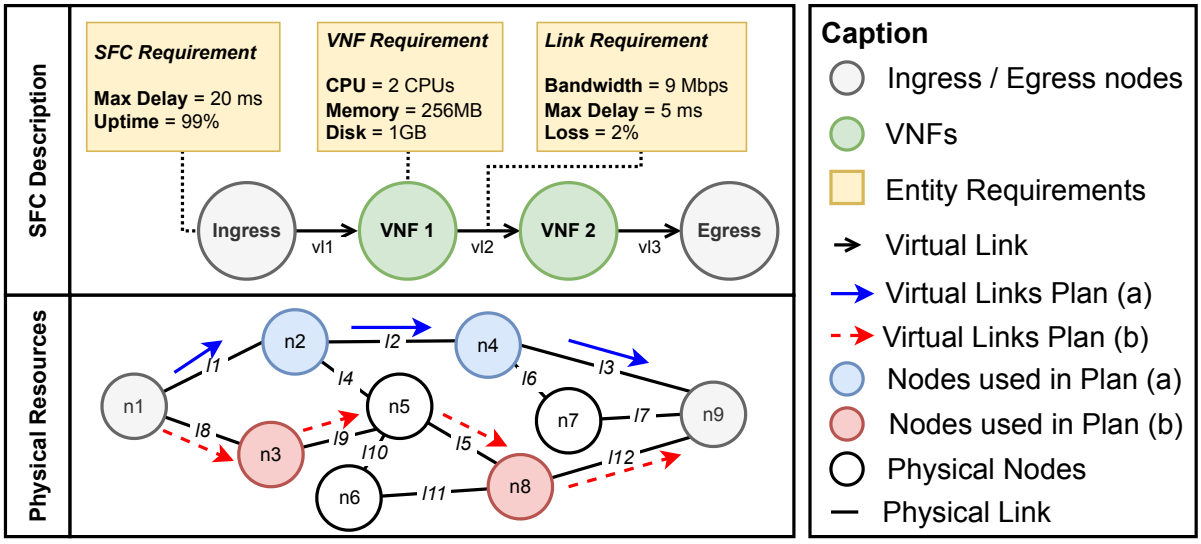


Figure 3: SFC elements and requirements overview.

where the user is associated and the domain where the elements of the SFC are executed belong to different administrative entities. In this multi-domain scenario, the domain owner would be reluctant to disclose details on their infrastructure to other infrastructure providers (TOUMI et al., 2021; ZHANG, J. et al., 2021) to enable the SFC execution.

Each SFC and VNF is characterized by its resource demand and requirements (MECHTRI et al., 2016). Typically, these resource demands are defined utilizing file descriptors with the TOSCA specification language (SOUSA et al., 2019). TOSCA allows the expressiveness of SFC and VNFs mappings data structures, providing methods for specifying workflows and enabling automatic lifecycle management tasks from the Management and Orchestration (MANO) tools, like the Open Source MANO (OSM).

The entities that compose the SFC are virtualized and executed on physical resources, such as edge or cloud nodes. Each component of the SFC, including the SFC itself, its VNFs, and the virtual links, has distinct requirements. Figure 3 illustrates an example of an SFC in which the maximum tolerated delay is 20 ms, and the execution of VNF 1 requires the physical node to reserve at least 256 MB of memory.

To execute an SFC, a MANO platform must be available (MAMUSHIANE et al., 2019). Different platforms can represent the SFC structure and requirements differently. Some of the most well-known SFC platforms are OSM[6], Cloudify[7], and Tacker[8] (MAMUSHIANE et al., 2019). A key component in the SFC platform is the NFV

[6]https://osm.etsi.org/
[7]https://cloudify.co/
[8]https://wiki.openstack.org/wiki/Tacker

Orchestrator (NFVO), also known as the NFV network service orchestrator. The NFVO is responsible for orchestrating NFVI resources across multiple Virtualized Infrastructure Manager (VIM) and also executes the lifecycle management component of network services to deliver end-to-end connectivity (SARRIGIANNIS et al., 2018; MAMUSHIANE et al., 2019).

When an SFC is initiated, its VNFs can be executed in either a single-domain or a multi-domain environment. The approach proposed in RFC 7665 defines that the elements of the SFC are executed within the same domain where the user is located (HALPERN et al., 2015). In contrast, a multi-domain strategy allows the elements of the SFC requested by the user to be executed across different administrative domains (TOUMI et al., 2021; LIU et al., 2020; MOHAMAD et al., 2019; ABUJODA et al., 2016). In this case, some or all components of the SFC are distributed among multiple domains. This approach can enhance the overall quality of service (QoS) delivered to the user (TOUMI et al., 2021).

2.4 SFC Placement Problem

In this section, we present the SFC Placement Problem (SFCPP). The SFCPP is the problem that, given an SFC, a set of conditions and restrictions, and an infrastructure that provides compute and network resources, finds the compute nodes to execute each VNF and the network links that provide connectivity through that SFC (KAUR et al., 2020; BHAMARE et al., 2016; ALAM et al., 2020). This problem was already proved as being an NP-Hard problem (KUO et al., 2018). Hence, to overcome the NP-Hard characteristic, multiple heuristics have been proposed to solve the SFCPP problem, (KIM et al., 2020; EMU et al., 2020; REYHANIAN et al., 2020; GAO, X. et al., 2022; AVASALCAI et al., 2019).

The objective of solving the SFCPP is to find a valid SFC Placement Plan (SFCPPlan). The SFCPPlan defines which physical resource will execute each element of the SFC (CISNEROS et al., 2022). Figure 3 also presents two possible ways to allocate elements of the requested SFC. In Plan (a), `VNF 1` is executed in node `n2` and `VNF 2` in `n4`. In Plan (b), `VNF 1` is executed in node `n3` and `VNF 2` in `n8`. A virtual link must also be associated with physical links. For example, in Plan (a) the virtual link `vl2` is mapped to the physical link `l2`, and in Plan (b) the virtual link `vl2` is mapped to a path of physical links `l9` and `l15`.

The SFCPP can be solved in a statically or a dynamically way (KAUR et al., 2020).

In the static approach, the placement plan is manually defined. In the work by Li et al. (LI, T. et al., 2017), they point out the following problems with the static approach: i) it is impossible to apply only desired network functions based on a specific flow, ii) Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) cost due to purchasing new hardware devices if the existing topology cannot fulfill the requirements, and iii) the network devices must be physically connected and manually configured by network operators, which may lead to inconsistent configurations.

In contrast, the dynamic approach creates the placement plan using algorithms. As highlighted in (KAUR et al., 2020; LI, T. et al., 2017), the dynamic approach offers several advantages, such as reduced CAPEX and OPEX as the controller steers the traffic over the environment, scalability, flexibility, and automatic service allocation, as well as the ability to only scale the virtual links when the traffic increases. However, a key disadvantage of the dynamic approach is that some algorithms take too long to produce the placement plan (LI, T. et al., 2017).

Solving the SFCPP using dynamic approaches has been widely studied and can be addressed through various strategies. These strategies are typically divided into two main categories: centralized and decentralized (or distributed) approaches. In centralized approaches, a single component has full knowledge of the physical infrastructure and generates the placement plan based on the requested SFC and the algorithm's logic (YI et al., 2018).

In decentralized approaches, the infrastructure knowledge is distributed among multiple components. Each component contributes to the creation of the placement plan based on its local view (CISNEROS et al., 2022). These approaches have gained momentum in recent years, especially with the growth of 5G networks and their demand for scalability and autonomy (HALABIAN, 2019).

2.5 Distributed SFC Placement Problem

One key component in any architecture that executes an SFC is the Management and Orchestration (MANO). MANO is responsible for coordinating the composition of multiple VNF that combined forms the SFC. By default, all the VNFs of an SFC are executed in the same domain (KUO et al., 2018). However, using only one domain to run the SFCs can struggle to deal with the restricted SFC requirements. Thus, new approaches like the multi-domain one must be addressed.

Typically, the works that address the SFCPP in a distributed fashion consider that the SFC is executed in a multi-domain environment (AVASALCAI et al., 2019), (CISNEROS et al., 2022), (SANTOS et al., 2022). A multi-domain environment can be defined as a set of independent entities with private computing and network resources that share limited information and can cooperate with each other to execute an SFC.

The work (CISNEROS et al., 2022) presents a multi-domain orchestration survey from an algorithmic perspective by considering cooperative and competitive approaches. They propose a taxonomy that considers network service life cycle tasks. Regarding privacy, they found that most works assume that the shared information does not compromise the privacy and security of the domain participant in the orchestration process. However, those works provide no guarantees to support this claim. The work (CISNEROS et al., 2022) also presents that the DSFCPP can be subdivided into three subproblems:

- **Shortest Path**: find the shortest path between the ingress and egress zone; if the delay in this path is higher than the max_delay, defined in the request, thus the problem does not have a solution;
- **Segmentation**: subdivide the VNFs of the SFC into VNF chunks;
- **Segment Assignment**: select the compute nodes in the shortest path that should execute each VNF chunk.

All the subproblems described above must be addressed to execute the SFC in a multi-domain environment. Each problem has security and time-consuming issues. In this work, we present a solution for each one of these problems in Chapter 4.

2.6 Game Theory

In this section, we briefly introduce the main concepts of Game Theory and its relation with the SFCPP. Game Theory is a theoretical framework for designing interaction among players who desire to achieve some goals (RASMUSEN, 2006). The focus of Game Theory is to provide tools for analyzing scenarios in which players make interdependent decisions to achieve their goals (OWEN, 2013).

Multiple classes of problems can be modeled using Game Theory (OWEN, 2013). One type of game was proposed by Rosenthal in 1973 (ROSENTHAL, 1973) named 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 2.1 formally describes a Congestion Game:

$$\Gamma = (N, R, (\Sigma_i)_{i\in[n]}, (d_r)_{r\in[R]}) \tag{2.1}$$

The elements of the Equation 2.1 can be defined as:

- 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 $d_r : \mathbb{N} \to \mathbb{R}$, 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 $r \in S_i$, in other words, the sum of the cost of multiple payers adopting the same strategy.

All congestion games have a Nash Equilibrium (NE) according to (ROSENTHAL, 1973). 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 (MASKIN, 1999; GAIRING et al., 2007). 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 a computationally complex task (GAIRING et al., 2007).

The complexity of reaching the NE rapidly increases as the number of resources and players grows. However, there is a subset of the Congestion Game named Singleton Congestion Game (SCG) (GAIRING et al., 2007), with lower computational complexity (MASKIN, 1999). 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 (SBABOU et al., 2012).

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 (BILÒ et al., 2017) investigate whether better performance is possible when restricted to the load-balancing problem in a game where players can only choose among single resources.

The SFC Placement Problem involves several steps, one of which is assigning each Virtual Network Function (VNF) to a specific compute node. To solve this challenge, our proposed approach maps the problem onto a SCG. Further details about this algorithm are provided in Section 4.5.3.

2.7 SFC Segmentation

In this section, we introduce the concept of SFC segmentation. The SFC Segmentation Problem consists of dividing an SFC into smaller parts, referred to as SFC segments, or simply segments (ABUJODA et al., 2016; SOUALAH et al., 2017). Each segment comprises a set of VNFs that must be executed within the same domain.

These segments consist of VNFs that are interconnected and operate together to perform specific network functions within the limits of their segment. Each segment is designed to handle a subset of the SFC VNFs, ensuring efficient and localized processing of network traffic. This segmentation facilitates better resource allocation, reduces latency, and improves the overall efficiency and manageability of network functions by leveraging localized optimizations and interconnecting segments through secure tunneling technologies such as VPNs or VXLANs (HUFF; VENÂNCIO; GARCIA et al., 2020).

To execute an SFC in a multi-domain, the chain of VNFs must be divided into segments. This segmentation process aims to balance resource utilization and ensure efficient coordination between domains. The main objectives of SFC segmentation are:

- **Resource Optimization:** Efficiently allocate computational and networking resources for each segment, ensuring that the VNFs operate within the constraints of their respective domains (XU, Y. et al., 2018);
- **Improved Manageability:** Smaller segments are easier to manage and orchestrate, especially in complex, multi-domain environments. This simplifies the overall management and monitoring of the SFC (MOUALLA et al., 2018);
- **Enhanced Security and Compliance:** SFC segmentation enables the isolation of specific VNFs to meet security and compliance requirements. This isolation ensures

that sensitive data flows through a secured path with the necessary inspection and enforcement points. By using secure tunneling technologies such as VPNs or VXLANs to interconnect segments in different network domains its possible to maintaining security and performance (HUFF; VENÂNCIO; GARCIA et al., 2020);

The SFC Segmentation problem, an essential aspect of the SFC Placement problem, remains relatively unexplored. Typically, studies incorporate the SFC Segmentation algorithm directly within the SFC Placement algorithm, such as (LIU et al., 2020). This tight coupling between the two problems can limit the efficiency of SFC Placement and restrict opportunities for advancement in the field of SFC Placement research.

The SFC Segmentation can be performed using a centralized or a decentralized approach. The centralized method is usually linked to centralized SFC Placement algorithms (LIU et al., 2020). In contrast, decentralizing the segmentation allows more effective segment selection in a distributed environment (PENTELAS et al., 2023).

A segmentation plan for an SFC consists of a set of segments and the VNFs assigned to each segment. The same SFC can have multiple segmentation plans, each tailored to meet different objectives or constraints. For example, if the goal is to minimize inter-domain network usage, a plan with fewer segments may be preferred. Figure 4 illustrates four possible segmentation plans for an SFC composed of three VNFs.

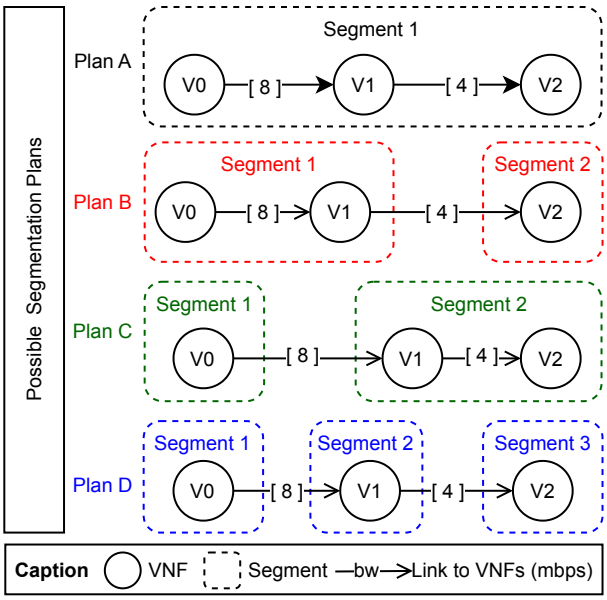


Figure 4: Example of SFC Segmentation Plans for an SFC with 3 VNFs.

SFC segmentation enables distributed processing and reduces inter-domain complexity, making it a keystone in the SFC placement process in multi-domain environments.

In the SFC placement process, only one SFC segmentation plan is selected for execution. This selection can optimize the placement to align with specific operational goals (PENTELAS et al., 2023).

2.8 Conclusion

In this Chapter, we explore the fundamental concepts of the NFV paradigm and NFV platforms, as well as the definition of SFCs and the importance of the SFC placement strategies. We also examine the role of Game Theory in addressing the SFC Placement Problem. Finally, we introduce the concept of SFC segmentation, which is a challenge that must be addressed to solve the SFC placement problem in a distributed fashion.

These presented concepts served as the foundation for our research. They guided the development of our new method to address the SFC Placement Problem using a distributed approach in a multi-domain environment. In the next chapter, we review related work and examine existing solutions proposed in the literature for this problem.

3 Related Work

This chapter presents studies directly related to our proposed approach. Section 3.1 analyzes algorithms that address the Distributed SFC Placement Problem (DSFCPP) in multi-domain environments. Section 3.2 explores SFC segmentation strategies adopted in other proposed solutions. Finally, Section 3.3 discusses architectures that enable the execution of SFCs across multiple domains. A comparison between these studies and our approach is also provided.

3.1 Distributed SFC Placement

In this section, we will present some relevant proposals that deal with the SFC Placement Problem (SFCPP). As explained in Section 2.4, 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 component with full knowledge of the network topology, domains, nodes 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 (SANTOS et al., 2022). The meaning of each column in the table is as follows:

- **Objective:** Identifyies whether the optimization problem is *mono-objective* (only one metric, such as cost or latency, is optimized) or *multi-objective* (two or more metrics are optimized simultaneously, requiring trade-offs);
- **Technique:** Describes the approach or method applied to address the problem, such as heuristics or exact algorithms;
- **Environment:** Presents the environment or context considered by the author, for example, Edge, Cloud, Edge-Cloud or not defined.

Table 1: Centralized SFC Placement approaches

Study	Objective	Technique	Environment
Ruiz et al. (RUIZ et al., 2020)	Mono	Genetic algorithm	Cloud
Kim et al. (KIM et al., 2020)	Mono	Conventional Lightchain algorithm	Not Defined
Emu et al. (EMU et al., 2020)	Mono	Machine Learning	Edge
Garrich et al. (GARRICH et al., 2020)	Mono	Greed	Edge
Li et al. (LI, D. et al., 2019)	Mono	Greed	Edge
Nguyen et al. (NGUYEN et al., 2020)	Multi	Markov Chain	Edge & Cloud
Reyhanian et al. (REYHANIAN et al., 2020)	Multi	Alternating Direction Method of Multipliers	Edge
Kiran et al. (KIRAN et al., 2020)	Multi	Genetic algorithm	Edge
Pei et al. (PEI et al., 2020)	Multi	Machine Learning	Not Defined
Bunyakitanon et al. (BUNYAKITANON et al., 2020)	Multi	Q-learning scheme	Edge
Pham et al. (PHAM et al., 2020)	Multi	Markov chain	Cloud
Li et al. (LI, S. et al., 2020)	Multi	Deep Reinforcement Learning	Cloud
Niu et al. (NIU et al., 2020)	Multi	Graph-based Particle Swarm Optimization	Edge
Alahmad et al. (ALAHMAD et al., 2020)	Multi	Greed	Cloud
Gao et al. (GAO, T. et al., 2020)	Multi	Steiner Tree and Markov Decision	Cloud
Mohamad et al. (MOHAMAD et al., 2019)	Multi	Greed	Edge

To overcome the issues related to centralized approaches, distributed approaches began to be proposed. In a systematic review, Santos et al. (SANTOS et al., 2022) 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 (VAQUERO et al., 2019; HUFF; VENÂNCIO; JR., 2019). Distributed approaches to solve the SFCPP were less common than centralized ones.

In most of the works reviewed in (SANTOS et al., 2022), the distributed aspect was

limited to the description of the experimental scenarios, rather than being applied directly to the SFC Placement problem itself. Typically, these works relied on a distributed framework that still included a component responsible for executing a centralized SFC Placement within each domain. Among the works surveyed, the only approach explicitly addressing SFC Placement in a distributed fashion followed a two-phase design: first, the VNFs of the SFC were divided into segments, and then each predefined segment was assigned to candidate domains to execute the VNFs within that segment.

Chen et al. (CHEN, C. et al., 2021) 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 in the selected domain path, the ingress orchestrator selects the next k-shortest path to execute the SFC. They also adopt the concept of VNF affinity and anti-affinity that allows the SFC Request to define if a VNF is forbidden or if it must be allocated in a certain domain.

However, the work of (CHEN, C. et al., 2021) 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, which will increase 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 medium/large scale networks. They also consider that the link requirements between all the 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, which

increases the number of rounds to completely allocate the SFC.

Another relevant work in this field was proposed by Liu et al. (LIU et al., 2020). 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 information about the border nodes and nodes where VNFs 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 will create 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 will use a distributed auction strategy to decide which domain will execute each segment.

Nonetheless, the work of (LIU et al., 2020) has some disadvantages. The adoption of a distributed auction strategy is typically employed in scenarios where the participants are competing against each other; however, in cooperative environments, like federations, where the objective is to maximize the general wealth rather than the individual gain, this strategy could not be applied. Another limitation is that distributed auction, according to (ZAVLANOS et al., 2008) 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 networks with a huge number of domains, the number of iterations can be considerable, making unfeasible the adoption of this strategy. Finally, they highlight that in the load balancing phase only domains excluded from the auction are allowed to compete. In other words, domains previously considered infeasible to execute the segment are reconsidered as candidates during load balancing. This creates a contradiction, since a domain identified as infeasible in the first phase is unlikely to become feasible in the second one.

Sun 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. All the domains will run the same internal placement component. The resources of each domain are independent, and the domain only shares the peering nodes and which VNFs it can execute. The algorithm executed by the main orchestrator is composed of two phases named partitioning and placement phase. The partitioning phase consists in 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. Nonetheless, in the innovations proposed in (SUN et al., 2018), the cornerstone resides in the full-mesh aggregation (LEE, 1995) approach to creating the topology of the aggregated graph. It means that direct virtual links must be created between all domains in the network. Therefore, in scenarios with a huge number of domains, this can consume valuable resources unnecessarily.

Gao et al. (GAO, X. et al., 2022) solve the placement problem in a collaborative edge and cloud computing environment. The environment is composed of low earth orbit satellites (edge nodes) and the cloud, which are organized in a hierarchical structure. The proposed approach is based on the Viterbi algorithm executed in each satellite. When the request arrives, the satellite shares with its neighbors the requested VNFs, and they decide if they can handle the request; if not, they offload the task to cloud nodes. Despite the collaborative behavior presented in this work, they do not consider the VNFs in the correct sequence, thus making the approach unfeasible to deal with SFCs as depicted by this work.

The approach proposed by (AVASALCAI et al., 2019) is based on a decentralized auction strategy. They seek to find 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 (SMT) to match each VNF with each domain. However, the proposed approach rapidly increases the computational demand in environments with more than 20 edge nodes, determining that this approach can only be used for finding a near-optimal solution in tiny environments.

Table 2 summarizes the presented distributed approaches and compares them to our proposal. The columns **Distributed Approach** briefly describe how the proposed solution handles the distributed placement, while the column **Segmentation** describes how the proposed solution builds the segmentation plan. There are three possible ways of segmenting the SFC: i) *No segmentation*, where all the VNFs are executed in the same compute node; ii) *Static*, where the segmentation is defined at the beginning of the placement process and does not change over time; and iii) *Dynamic*, where the segmentation process is defined recursively during the creation of the placement plan.

Table 3 presents a comparison between the Game Theory-based approach and other commonly used techniques for SFC placement. It highlights the main limitations

Table 2: Distributed SFC Placement approaches

Study	Distributed Approach	Segmentation
Chen et al. (CHEN, C. et al., 2021)	The allocation of each VNF segment in each domain is executed in parallel.	Static
Liu et al. (LIU et al., 2020)	The auction consensus phase defines which segment will be executed in each domain.	Static
Sun et al. (SUN et al., 2018)	The allocation of each sub-SFC in each domain is executed in parallel.	Static
Avasalcai et al. (AVASALCAI et al., 2019)	Auction-based with a centralized orchestrator that defines the winner domain for each VNF in the SFC.	Static
Gao et al. (GAO, X. et al., 2022)	The allocation of each service in each satellite is executed in parallel.	No Segments
Zhang et al. (ZHANG, Z. et al., 2024)	Joint SFC deployment problem for Edge-Cloud networks with the goal of maximizing network.	Static
SPEED	Create segments while executing Games to find a suitable domain for each VNF.	Dynamic

of alternative methods that are effectively addressed by game-theoretic modeling, particularly in distributed and multi-domain environments. The table demonstrates how Game Theory offers advantages in scalability, privacy preservation, and decentralized coordination, making it suited for realistic and dynamic scenarios.

Table 3: Comparison of Game Theory with other SFC Placement techniques

Technique	Limitations overcame by Game Theory
Auction-based	High communication complexity; potential for non-cooperative behavior; slow convergence in large networks.
Genetic Algorithms	Require global knowledge; difficult to distribute; high computational cost; solutions are not guaranteed to be stable.
Machine Learning / Reinforcement Learning	Depend on extensive training and global data; lack of transparency in decision-making; risk of overfitting.
Greedy / Viterbi / Heuristic-based	Fast but not robust; poor adaptability to dynamic environments; limited in achieving globally optimal solutions.
SMT / ILP / Dynamic Programming	Computationally infeasible at large scale; require complete topology and constraint knowledge; not suitable for real-time execution.

Despite some advances brought by Game Theory in addressing the challenges of SFC placement, certain limitations still arise when applying this approach. These limitations include computational overhead and model complexity. As shown in Table 3, alternative

techniques such as auction-based methods, genetic algorithms, and reinforcement learning also face significant drawbacks, including high communication costs, the need for global knowledge, and lack of transparency in decision-making. The design of SFC placement mechanisms based on Game Theory must explicitly account for these challenges to ensure practical solutions.

3.2 SFC Segmentation

In this section, we discuss how different SFC placement methods approach the SFC segmentation problem. Segmenting the SFC is a required step that enables its execution across multiple domains. Independent of the chosen placement strategy, this step is essential to ensure the placement feasibility in a multi-domain environment.

In (PENTELAS et al., 2023), the SFC segmentation problem is solved by developing a decentralized framework that assigns independent Double Deep Q-Learning (DDQL) agents to each Virtualized Infrastructure Manager (VIM). Those agents make local decisions based on detailed observations and minimal exchange of critical information with neighboring agents. That approach improves scalability and resource allocation efficiency. An action coordination module at the NFVO layer aggregates those local decisions into a globally optimized solution.

The authors of (RODIS et al., 2023) address the SFC segmentation problem by proposing a Genetic Algorithm (GA) based solution that dynamically adjusts parameters to adapt to various network topologies and SFC requests, ensuring efficient resource utilization and minimizing segmentation. It uses heuristics for initial population generation to guide the search effectively, optimizing the Cost-to-Revenue Ratio (CRR) for better resource efficiency. The GA-PAGA variant reduces inter-rack traffic by colocating VNFs within the same rack, which lowers latency and bandwidth usage. In general, the solution balances computational load and scalability, making it suitable for large-scale network applications.

In (LIU et al., 2020), the partitioning phase of the GDM (General Distributed Method) for cross-domain SFC placement involves several key steps to efficiently divide the SFC into segments and assign them to suitable domains. Initially, the receiver domain traverses the SFC, creating segments based on VNFs location constraints. Domains then bid for these segments using a decentralized auction mechanism, calculating bid values with a utility function tailored to their placement goals. Through a consensus-

based segment auction (CSA), domains iteratively exchange and update local winning bid lists until a stable segment assignment is achieved. Following this, inter-domain paths are selected to connect the segments, ensuring minimal additional costs and meeting bandwidth requirements. If any segment cannot be assigned due to resource constraints, the algorithm re-divides and re-assigns the segments until all possible placement solutions are considered, thereby maximizing the acceptance ratio and ensuring efficient resource utilization across the substrate network.

The studies cited previously show that the placement of segments in a multi-domain environment is a complex and time-consuming task. Thus, adopting a strategy to choose a suitable segmentation plan with respect to the environment is necessary to reduce the time to place the requested SFC (LIU et al., 2020). In the literature, segments are typically defined during the initialization of the placement process and do not change during the placement phase. In Section 4.5.2.2 we will present our proposed approach to solve this problem in a distributed fashion.

3.3 Architectures for Distributed Placement of SFCs

This section presents work that proposed architectures or frameworks that support distributed solutions to the SFC Placement Problem (SFCPP). As it is a new topic, few researchers have addressed it. Table 4 depicts the comparison between the architectures.

Huff et al. (HUFF; VENÂNCIO; JR., 2019) extended the European Telecommunications Standards Institute model proposing the Multi-SFC Orchestrator, depicted in Figure 5. Figure 6 depicts the proposed architecture, which allows the deployment of VNFs that compose an SFC among multiple domains. Each domain can execute different orchestrators and adopt different network topologies and technologies. Inside the domain, there are *NFVO Agents* and *VIM Agents*. Those components receive commands from the main orchestrator, named Multi-SFC Orchestrator, and translate to the local domain NFV Orchestrator and Virtualized Infrastructure Manager. They are drivers that enable the orchestrators and Virtualized Infrastructure Manager heterogeneity. The forwarding of flows between segments, for instance, from $segment1$ in $domain1$ to $segment2$ in $domain2$ is achieved through Virtual Private Network (VPN). After the last VNF of $segment1$ and before the first VNF of $segment2$, additional VNFs are deployed to run the VPN components. These components interconnect the domains, allowing the SFC flow to traverse them seamlessly. However, the text does not formally define how the

segmentation process is managed within the Multi-SFC Orchestrator. Moreover, since this orchestrator is a single component in the environment, it introduces a potential single point of failure.

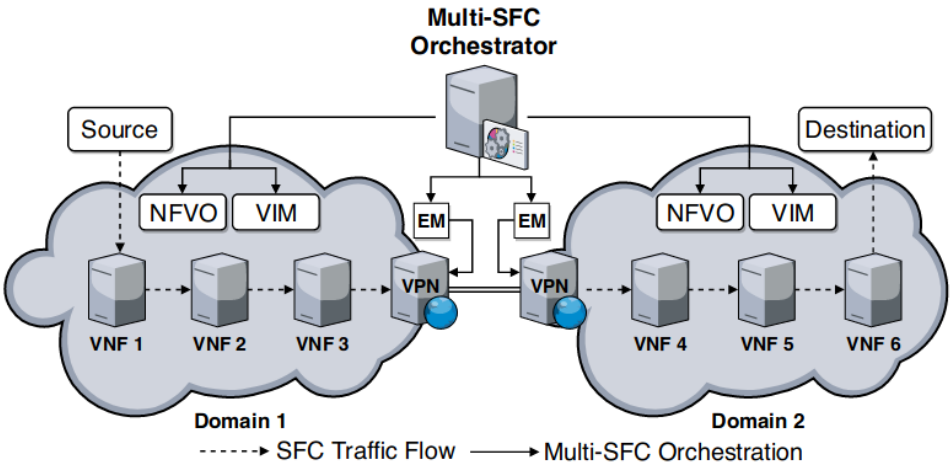


Figure 5: Multi-domain approach (HUFF; VENÂNCIO; JR., 2019).

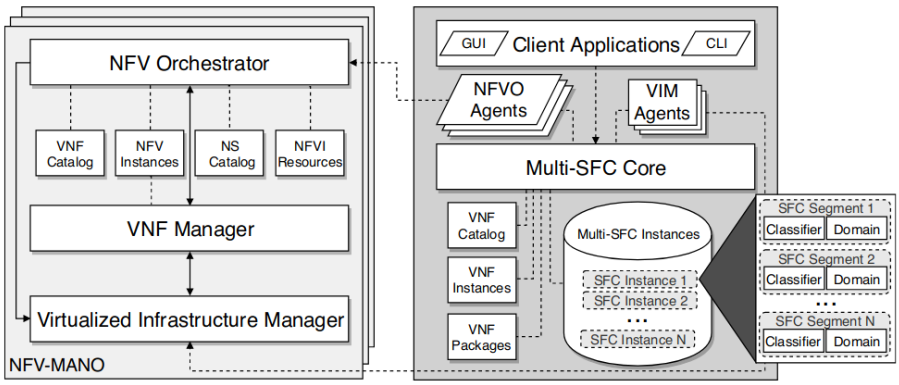


Figure 6: Multi-domain architecture (HUFF; VENÂNCIO; JR., 2019).

The work firstly proposed by Toumi et al. in (TOUMI et al., 2020) and extended in (TOUMI et al., 2021) presents an orchestration architecture that leverages existing standardization efforts. It adopts the TOSCA language to specify the network services and VNF Descriptor (TOSCA, 2017), in order to ensure an end-to-end orchestration of SFCs in a multi-domain environment. Figure 7 depicts the architecture proposed in (TOUMI et al., 2021). Their proposal works regardless of the internal domain communication protocols. They also propose a multi-domain SFC deployment protocol that configures the network components to meet the delay. Despite proposing an SFC Partitioning workflow, they do not specify how the segments are created.

Table 4: Comparison of Multi-domain SFC Orchestration Architectures

Feature	(HUFF; VENÂNCIO; GARCIA et al., 2020)	(TOUMI et al., 2021)	**SPEED**
Architecture type	Centralized with Multi-SFC Orchestrator	Centralized orchestration using TOSCA language	Distributed with orchestrator being selected based on the SFC request
Support for multiple domains	Yes, with VPN-based interconnection between domains	Yes, independent of internal domain communication protocols	Yes, Using an hierarchical approach.
Inter-domain communication	VPN components inserted between segments	Custom multi-domain SFC deployment protocol	Custom SFC deployment protocol
SFC segmentation approach	Segmentation not formally defined	Workflow proposed, but segment creation not detailed	Distributed SFC Segmentation
Single point of failure	Yes, due to the centralized Multi-SFC Orchestrator	Yes, although standardized, orchestration is still centralized	No, any domain can act as an orchestrator

3.4 Conclusion

In this chapter, we reviewed the main studies related to the SFC Placement Problem in multi-domain environments. We particularly focus on distributed approaches and segmentation strategies. We examined how different methods address the challenge of segmenting SFCs, which is a key step to enable their deployment across multi-domain environments.

We also analyzed existing architectures that support distributed placement, noting that most of them still rely on a central orchestrator. Another important observation is that the existing platforms do not clearly define how segmentation should be performed. Finally, we highlighted key limitations of current solutions, such as static segmentation, lack of scalability, and the presence of single points of failure. These limitations reinforce

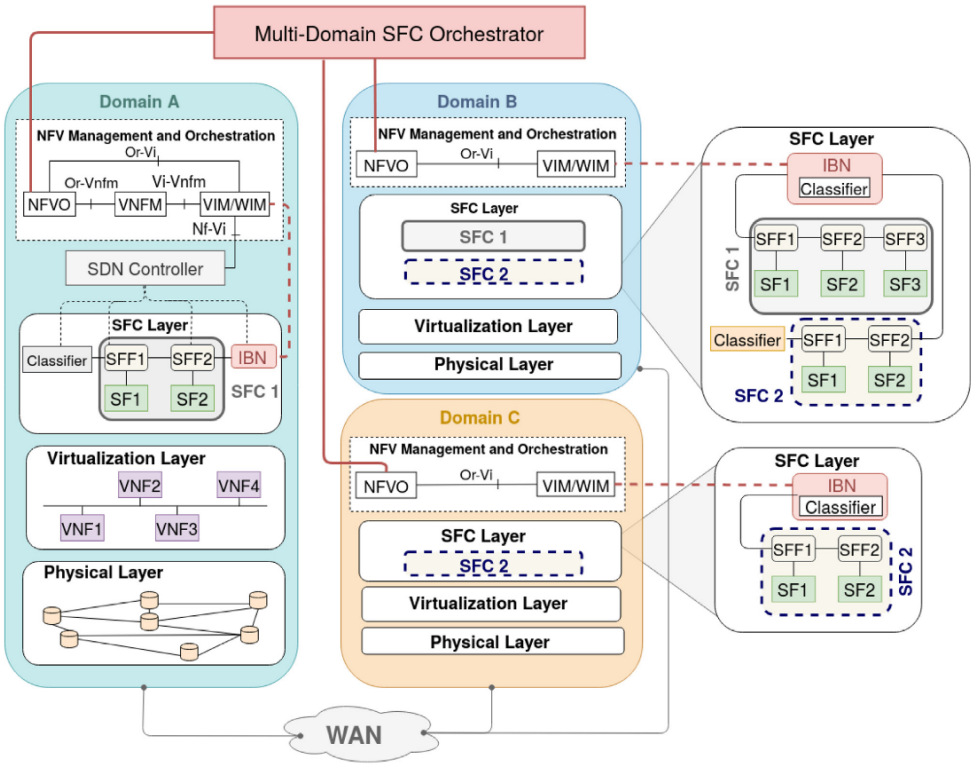


Figure 7: Multi-domain Architecture proposed by (TOUMI et al., 2021).

the need for a more dynamic and fully distributed approach to SFC placement.

The gaps identified in this chapter serve as motivation for the proposal that will be presented in Chapter 4. In that chapter, we will describe in detail the environment in which the proposed solution operates. Additionally, the algorithms designed to perform SFC placement in a distributed environment will be introduced, using Game Theory to support the decision-making process.

4 SPEED - SFC Placement in Edge-Cloud Continuum

This chapter presents our proposed approach, named SFC Placement in Edge-Cloud Continuum (SPEED). We summarize the proposal in Section 4.1. The definition of how the topology will be hierarchically arranged in the distributed environment is depicted in Section 4.2. After that, the architectural components are discussed in Section 4.3. Then, in Section 4.4, we describe the system model and the problem formulation. After that, we portray in Section 4.5 the heuristics for each step of the proposed solution. Lastly, Section 4.6 describes the operational workflow of the system, including how data aggregation and service placement requests are handled by the proposed approach.

4.1 Core Principles

This section provides an overview of SPEED, a proposal designed to solve the SFC Placement Problem (SFCPP) using a distributed approach. The solution aims to find a suitable placement plan, if one exists, for a requested SFC, considering its requirements and the resources available in the environment. Our approach is based on three core pillars:

1. **Hierarchical Topology**: A virtual structure that organizes all entities and available resources in a tree-like topology;
2. **Data Aggregation**: The process of collecting and propagating metadata about the VNF Types available in each domain throughout the hierarchical topology;
3. **Distributed Placement**: The set of algorithms and components responsible for determining a suitable placement plan in a distributed manner, leveraging both the hierarchical structure and the aggregated data.

Our approach differs from other strategies presented in the literature based on how the VNFs are assigned to each compute zone. In SPEED, the zone that initiates the

placement process does not have prior knowledge of where the VNFs will be allocated across compute zones. Instead, the assignment of each VNF is determined on the fly in a distributed manner, as the decision process unfolds. The detailed procedures for this process are discussed in Sections 4.5.2 and 4.5.3.

We divided the proposed solution into multiple steps, each supported by a heuristic that addresses the necessary aspects of the process. These steps are presented in Section 4.5, along with detailed architectural components that illustrate the elements required to deploy the solution in real-world environments. Additionally, the system operations needed to execute our approach are described in the subsequent sections.

4.2 Hierarchical Organization and Data Aggregation

This section presents the hierarchical organization and the data aggregation process. The hierarchical organization defines how we mapped the physical hierarchical structure into an abstracted hierarchical structure. The data aggregation process defines how the meta data is processed over the abstracted hierarchical structure.

4.2.1 Hierarchical Organization

This section presents the environment where SPEED is executed. The environment is composed of zones. A zone is an abstraction used in this work to identify entities that provide computing and network resources or aggregated data. These entities can be edge or cloud data centers. The physical hierarchical structure is mapped into zones, which are hierarchically organized. Each computational unit (data center, edge node, etc.) will be mapped to a compute zone, which will be organized hierarchically regarding the zone owner's preferences.

Figure 8 depicts an abstract view that encompasses i) zone types and ii) hierarchical topology. There are three types of zones, namely Access, Compute, and Aggregation. Each type of zone has a specific meaning in the proposed approach as described below.

Access zones provide network connectivity to end users. In a 5G scenario, these zones correspond to the components of the Radio Access Network (RAN). Multiple access zones may be associated with the same parent zone. VNFs are not executed on access zones.

Compute zones are the elements that provide computational resources. They can

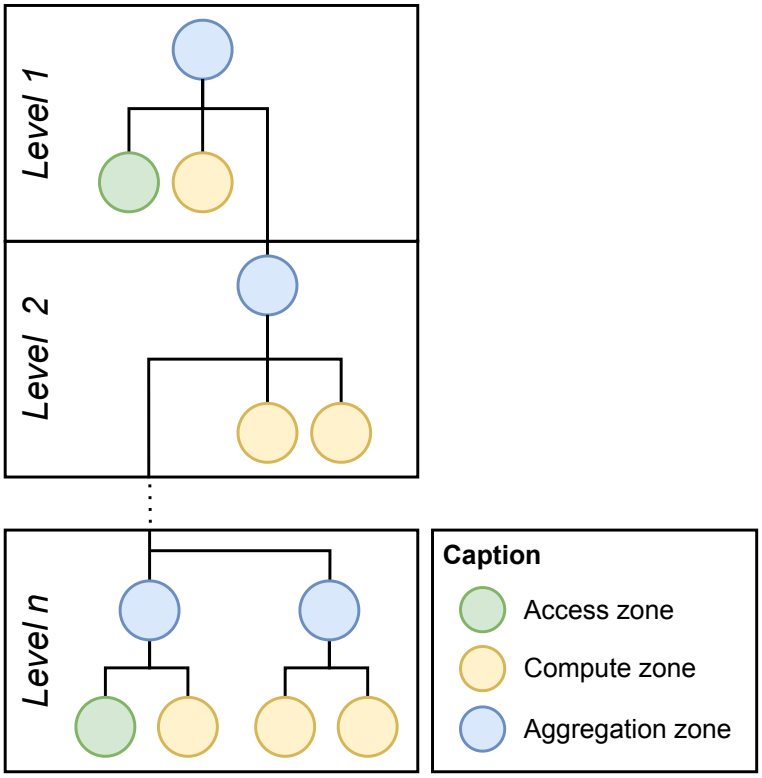


Figure 8: Environment Topology Overview.

represent either edge or cloud data centers. VNFs are executed on the Compute zone nodes. Various VNF Types can be executed in each Compute zone.

Aggregation zones are abstracted elements adopted in our approach that aggregate data about the descending zones.

Figure 9 depicts an example of how an organization is portrayed in a hierarchy structured based on our proposed approach. Figure 9 (A) illustrates a University hierarchy comprising Campuses, Departments, and Courses. The entities "University A", "Campus B", "Campus C", "Department E" and "Course I" have their own data centers (cloud or edge). Figure 9 (B) depicts how the physical structure is mapped into the zone hierarchy proposed in this thesis.

University A has a data center that can be used to execute VNFs, which is represented by the compute zone A'. Furthermore, University A can execute VNFs in one of its subordinated entities, such as campuses, departments, and courses. To represent this, an aggregation zone A is created which aggregates the data of the entities beneath it. For example, Campus C only has one data center, so in the hierarchical structure, there is only one compute zone mapped and associated with aggregation zone A.

The distributed environment is arranged in a hierarchical structure, with one

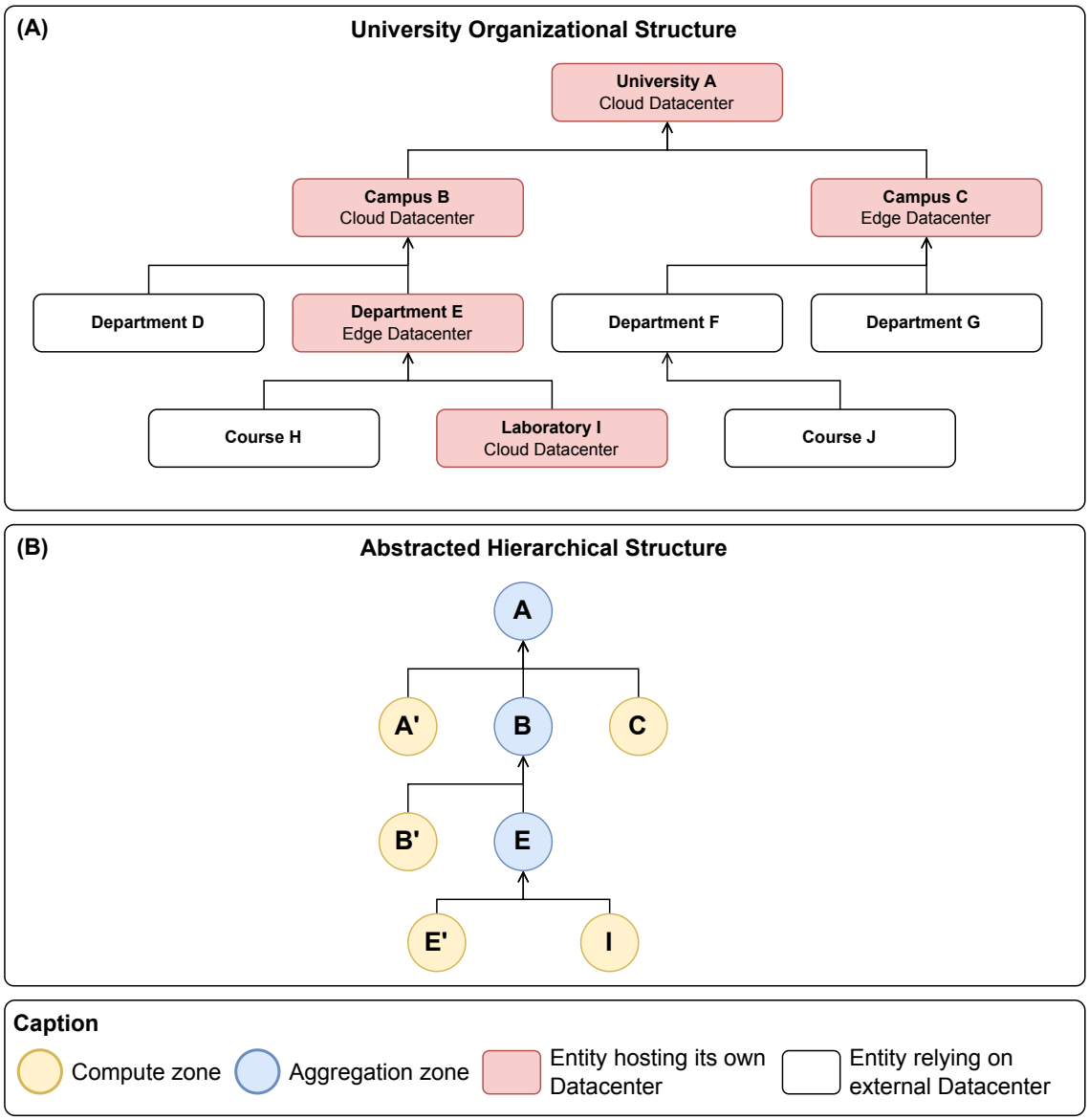


Figure 9: Mapping a physical hierarchy into our proposed abstract hierarchy.

Aggregation zone at the top and the other zones subordinated to it. This tree-like topology determines how the aggregate data is managed, allowing for efficient data sharing across the entire system. The higher-level zone is the only one that is not subordinated to anybody.

The hierarchical organization presented in this section is a flexible model for structuring computing and network resources in distributed environments. Its configuration may vary depending on how the system administrator groups compute and aggregation zones, considering factors such as performance, latency, or organizational policies. This adaptability makes this hierarchical organization suitable for a wide range of contexts, including edge–cloud 5G scenarios.

4.2.2 Data Aggregation

In this section, we describe how metadata flows within the environment based on the proposed solution. As introduced in Section 2.5, the Edge-Cloud continuum involves a large number of compute nodes and network links. This scale makes the deployment of VNFs across the infrastructure particularly complex.

Centralized orchestrator-based approaches, such as (SUN et al., 2018), require a complete view of the topology on a single node. This introduces technological and business model limitations. As noted in (SANTOS et al., 2022), such centralized solutions are difficult to adopt in real-world environments.

To overcome these limitations, our approach stores partial and aggregated metadata instead of the full topology. This strategy has been effectively used in other domains, such as hierarchical routing (LEE, 1995). We apply the same principle to enable distributed SFC placement across multiple domains.

In our approach, the **Aggregation zones** process and store all the data sent by their underlying child zones. Compute, Access and Aggregation zones send their internal data solely to the Aggregation zone of which they are child. The metadata aggregated data flow from the bottom to the top-level Aggregation zones. The top-level Aggregation zones have aggregated information about their own Compute zone and all the bottom-level Aggregation zones.

By adopting data aggregation strategies, the Aggregation zone will provide partial information to its parent zone. The aggregated data in each Aggregation zone is created using a combination of i) the data sent by the underlying Aggregate and Compute zones, ii) the data already stored in the Aggregation zone, and iii) data gathered by the Aggregation zone, like network status and services status from its child zones. Table 5 outlines which raw data type the underlying zone will provide to the parent zone.

The Aggregation zone updates the aggregated data in response to the changes reported by the child zones. Furthermore, the Aggregation zone can also update the aggregated data when a child zone fails to execute the request for executing a set of VNFs. When the aggregated data is updated, the Aggregation zone will inform the parent zone about its new status.

The child zones will send data to their parent zone when the zone is configured in the environment or when the data previously sent becomes invalid. The data stored in the zone became invalid in the following situations: i) when a previously available VNF

Table 5: Data aggregation fields.

Name	Description
Zone	The name of the zone. The zone can be an **Aggregation zone** or a **Compute zone**.
VNF	Defines the VNF Types that can be executed in the zone.
Gateway	The Gateway where the packets will flow from to access the zone.
Cost	The cost to execute the VNF in the zone.
Delay	For a **Compute zone**, the delay is based on the network the propagation delay from the Gateway to the zone. For an **Aggregation zone**, the delay is the sum of propagation delays from the Gateway to the zone plus the time already consumed between the Aggregation zone and the child zone where a VNF will be executed.

can no longer be executed in the zone, ii) when a VNF previously unavailable became available in the zone, and iii) when a new VNF type became available. The information about the available resources in each physical node inside a Compute zone will not be sent to the parent zone. Only information about each VNF that can be executed in the zone will be sent. This approach of not sharing restricted information about the node resources and topology increases data privacy, thus improving the security of the overall business model and infrastructure (XU, Q. et al., 2018) in our solution.

Figure 10 depicts an example of the environment and the data aggregation over time. The environment comprises 3 Aggregation zones, named `A1`, `A2`, and `A3`. The Aggregation zone `A2` has two child zones, the Compute zone `C1` and `C2`. The Compute zones inside `A1` and `A2`, besides all the Access zones, will not be depicted in the figure to increase diagram plainness.

The data stored inside each zone, either Compute or Aggregation, changes over time. The Aggregation zone `A2` stores aggregated data from `C1` and `C2`. In time `T0`, `A2` stores the aggregated information that the best Compute zone to execute a VNF of type `4` is the Compute zone `C2`. In time `T1`, `C2` informs `A2` that the VNF of type `4` can not be executed anymore; thus, `A2` updates its aggregated data removing the possibility of VNFs of type `4` being executed because none of `A2` child zones can execute this VNF anymore. In time `T2`, Compute zones `C1` and `C2` will inform `A2` that they can execute VNFs of type `4`, and `A2` updates the aggregated data pointing that the Compute zone `C1` now is the best zone to execute VNFs of type `4` because the delay to Gateway 1 is lower in `C1` than `C2`.

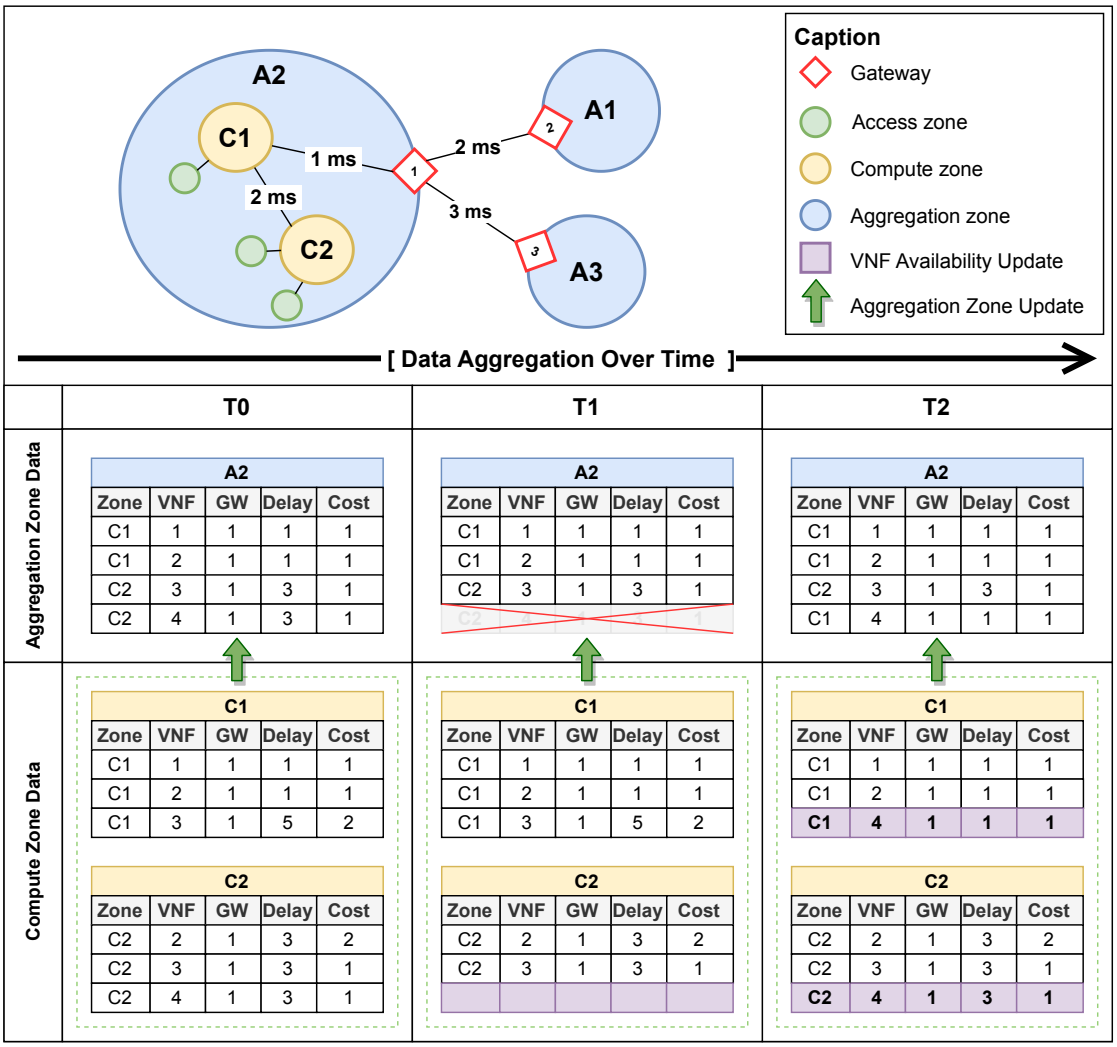


Figure 10: Data aggregation in a distributed environment over time.

4.3 Architectural Components

This section presents the architecture that enables the proposed SPEED approach, which extends the components and interface specifications from ETSI (ETSI, 2022). Figure 11 depicts the architectural components. Inside each Aggregation zone, previously presented in Figure 8, there is an NFVO-MANO and the SPEED components. The NFVO-MANO executes the VNF management and orchestration tasks. The SPEED component will coordinate the zones to execute the VNFs required by each SFC.

The proposed architecture enables distributed SFC placement across multiple domains. Each domain operates with shared objectives, such as minimizing placement costs. As a result, even when acting independently, all domains cooperate, either explicitly or implicitly, to collectively achieve the global optimization goals.

The *Network Zone Manager* subsystem is responsible for managing the relationship

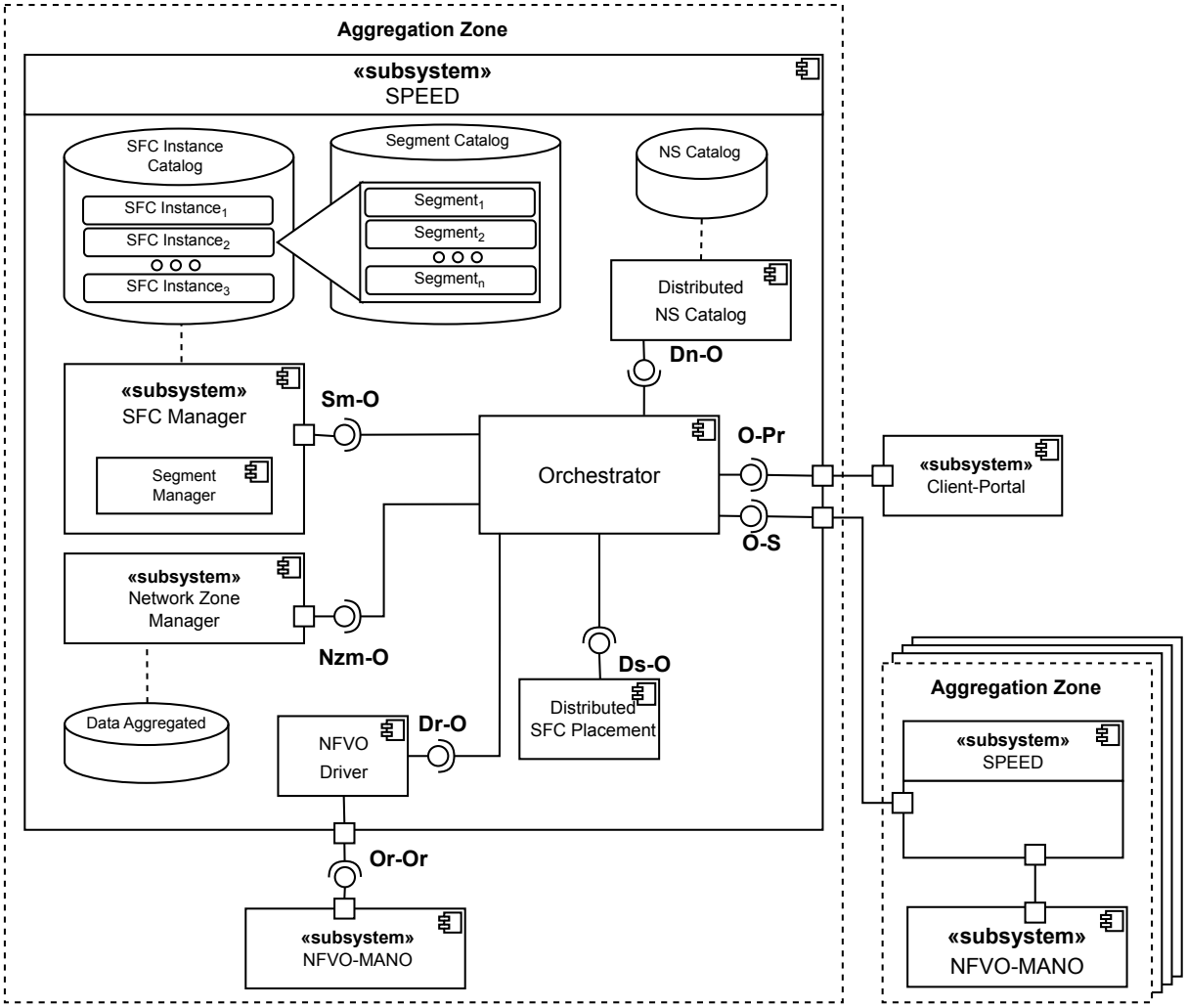


Figure 11: Architecture components of our approach.

between the zones. The zone hierarchy must be defined by the zone owner using some heuristic. In this work, we do not cover this environment coordination aspect. This subsystem provides the *Nzm-O* interface that allows the orchestrator to collect information about the zones. Table 6 presents the functionalities provided by the interface.

Table 6: *Nzm-O* Interface Capabilities

Function	**Description**
getComputeZones()	Return a list with all the child compute zones.
getAggregationZones()	Return a list with all the child aggregation zones.
updateParentZone(zone)	Change the parent zone.

The *Distributed NS Catalog* component is responsible for managing the descriptors of the SFCs that can be executed in the zone. It comprises the descriptor of SFCs that can be fully executed within the zone, as well as those that depend on inter-domain coordination for execution in the zone and the SFCs that need coordination between zones

to be executed. It provides the *Dn-O* interface that allows the orchestrator component to manage which SFCs can be executed in the zone. Table 7 presents the capabilities offered by the interface.

Table 7: *Dn-O* Interface Capabilities

Function	**Description**
getSFCTypeAvaibale()	Return a list with all available SFC Types.
getVNFTypeAvaibale()	Return a list with all available VNF Types.

The *NFVO Driver* component is responsible for translating the commands from the Orchestrator to the NFVO-MANO. There will be one *NFVO Driver* for each type of NFVO-MANO available. As each zone has its own NFVO-MANO, this strategy allows the interoperability of our proposal in heterogeneous environments. The *NFVO Drive* installed in a zone must be compatible with the NFVO-MANO configured in the zone. It provides the *Dr-O* interface that allows the Orchestrator component to communicate with the NFVO-MANO in the zone using the interface *Or-Or*. Table 8 presents the capabilities offered by the interface.

Table 8: *Dr-O* Interface Capabilities

Function	**Description**
executeSFC()	Request the creation of a new SFC.
statusSFC()	Request the status of SFC in execution.
deleteSFC()	Request the SFC to be deleted.

The *SFC Manager* component is responsible for managing the SFC Instances lifecycle. When the zone is defined as the Manager Zone of an SFC request, the SFC Instance metadata associated is created in this zone component. This component provides the *Sm-O* interface that allows the orchestrator component to manage the SFC Instances. Table 9 presents the capabilities offered by the interface.

Table 9: *Sm-O* Interface Capabilities

Function	**Description**
saveSFCMetadada()	Save the SFC Instance metadada.
deleteSFCMetadata()	Delete the SFC Instance metadada.
getSFCMetadata()	Return all the metadata from an SFC Instance.

The *SFC Instance Catalog* and *Segment Catalog* store data about the segments executed in a child zone. Each segment is associated with one SFC Instance. The SFC Manager component consumes and updates data in both storages.

The *Orchestrator* component is responsible for receiving the SFC request from the Client-Portal via *O-Pr* interface. The client portal is accessed by the user that requests the creation of new SFC Instances. Table 10 presents the capabilities offered by the interface.

Table 10: *O-Pr* Interface Capabilities

Function	**Description**
requestSFC()	Request the creation of a new SFC by the user.
statusSFC()	Request the status of an SFC Instance by the user.
deleteSFC()	Request to delete an SFC in execution.

The *Orchestrator* component also provides the *O-S* interface that allows the orchestrator component to communicate with the SPEED component executed in another aggregation zone. This interface allows the required coordination between zones to execute the SFC Instance in a multi-domain environment using a distributed fashion approach. Table 11 presents the capabilities offered by the interface.

Table 11: *O-S* Interface Capabilities

Function	**Description**
aggregatedData()	Receive the aggregated data from another zone.
SFCRequest()	Receive a request to execute an SFC in the domain.

The *Distributed SFC Placement* provides the *DS-O* interface that allows the orchestrator to request the execution of the SFC Placement in a distributed fashion. This component is responsible to play the game to determine the zones that will execute each VNF in the SFC. Table 12 presents the capabilities offered by the interface.

Table 12: *DS-O* Interface Capabilities

Function	**Description**
executeGame()	Execute the algorithm that determines which zone will execute each requested VNF.

In this section, we presented the architecture in which the proposed solution will be executed. It enables communication between zones to aggregate metadata about the available VNFs and SFC types in the environment. In the next section, we will present the system model for the SFC placement problem in a distributed manner.

4.4 System Model

In this section we present our system model. We modeled our SFC Placement as an online problem. The SFC requests are processed individually when each one arrives in the environment. The problem was elaborated as a Mixed-Integer Programming (MIP) model.

In the 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.

We modeled the environment composed of edge and cloud data centers. The VNFs of the SFC request are executed both in edge or cloud nodes according to the resources available and restrictions imposed by business rules, user restrictions, and request needs. Table 13 depicts all the symbols and decision variables of the proposed system model.

The network is modeled as an undirected graph $G = (\Omega, L)$. 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 $parent == NULL$ 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 (4.2) 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 physical link l_j.

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

Table 13: System Model notation

Notation	Description
Parameter Variables	
$G = (\Omega, L)$	Undirected graph of the physical network.
Ω	The set of network zones.
L	The physical links between the zones.
Z	Set of all the VNFs in the environment.
$\Re$	The set of SFC requests.
r	The SFC request definition.
src	The node source of the dataflow node.
dst	The node destination of the dataflow.
V	The VFN that composes the requested SFC.
VL	The Virtual Links that bind the VNFs.
vl	The Virtual Link that connects two VNFs.
max_delay	The maximum tolerable network delay between two VNFs.
$childrens_{\omega}$	List with the children's zones of the zone ω.
$parent_{\omega}$	The parent zone of ω.
τ	The time when the SFC request arrives in the environment.
Λ_{ω}	Set of VNFs that can be executed by at least one zone ω subordinate.
cpu_z	The amount of CPU required to run a VNF.
mem_z	The amount of memory required to run a VNF.
bw	The required network bandwidth.
max_delay	The maximum tolerable delay of an SFC.
ω	Define a single zone.
Φ	Identify whether a VNF is permitted or prohibited for deployment in a specific zone.
κ_{ω}	The type of zone, binary variable (Compute Zone = 1, otherwise 0).
$CompCost$	The computing cost to execute the VNFs.
$NetCost$	The cost of the network to handle the traffic.
Decision Variables	
x_{ij}	Defines if VNF n_i will be executed in zone ω_j.
y_{ij}	Defines if virtual link v_i will be executed in link l_j.

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 Equation (4.1), 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.

$$\Phi_{\omega_i,\omega_j,z} = \begin{cases} 1, & \omega_i \text{ allows VNF } z \text{ in } \omega_j \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

$$\kappa_{\omega} = \begin{cases} 1, & ComputeZone \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

The set $L = \{l_1, l_2, ..., l_n\}$ represents the links between the zones. For every link $l_i \in L$, between two adjacent zones ω_1 and ω_2, we use bw_l and $delay_l$ to denote its bandwidth capacity and delay respectively. The parameter Ψ_l^v represents the cost of executing the virtual link v in the physical link l. Links between sibling compute zones have infinity *bw*, *delay*, and $cost = 0$.

4.4.1 Problem Formulation

In this section, the SFC Placement Problem (SFCPP) is formulated. In the previous section, we introduced the decision variables of the model, shown in Table 13, and the interdependencies between them. In this section, we discuss the constraints and objective functions of the model. The objectives aim at minimizing: i) the cost of computational resources, ii) the cost of network resources, and iii) the SFC network delay.

The SFC requests, after being placed, will consume computational and network resources. The service provider will charge a fee to execute the SFC request, according to the consumed resources. The cost of the services will depend on the total computational resources used in edge and cloud nodes, besides the network bandwidth used in the links, and the traffic routing through the VNF chain.

Thus, our problem formulation requires two decision variables. The decision variable x_{ij} that defines if VNF n_i will be executed in zone ω_j and the decision variable y_{ij} that defines if virtual link v_i will be executed in link l_j.

Thus, given a set of SFC requests, our goal is to minimize the overall cost. The total placement cost, named $PlacementCost$, consists of two components: the computational resource cost, named $CompCost$, and the network resource cost, named $NetCost$.

$$PlacementCost = CompCost + NetCost \tag{4.3}$$

Equation 4.4 defines how the computational resource cost is computed.

$$CompCost = \sum_{z_i \in Z} \sum_{\omega_j \in \Omega} x_{ij} * \beta_{\omega_j}^{z_i} \tag{4.4}$$

Equation 4.5 defines how the network resource cost is computed.

$$NetCost = \sum_{v_i \in V} \sum_{l_j \in L} y_{ij} * \Psi_{l_j}^{v_i} \tag{4.5}$$

Given the zones, links, and resources available, the objective function defined in Function 4.6 is to find the placement plan for the VNFs and the subsequent traffic route that minimizes the deployment cost.

$$minimize\ PlacementCost \tag{4.6}$$

The problem is subject to some constraints. Equation (4.7) defines that the VNF can only be executed in Compute Zones. Equation (4.8) defines that the VNF can only be executed in zone ω when the VNF is available in that zone. Equation (4.9) defines that the VNF can only be executed in zone ω when there are no affinity (Φ) restrictions. These equations are placement 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.

$$\sum_{z_i \in Z} \sum_{\omega_j \in \Omega} x_{ij} * \kappa_{\omega_j} \geq 1 \tag{4.7}$$

$$\sum_{z_i \in Z} \sum_{\omega_j \in \Omega} x_{ij} * \Lambda_{\omega_j}^{z_i} \geq 1 \tag{4.8}$$

$$\sum_{z_i \in Z} \sum_{\omega_j \in \Omega} \sum_{\omega_k \in \Omega} x_{ij} * \Phi_{\omega_j, \omega_k, x_{ij}} \geq 1 \tag{4.9}$$

In this section, we presented the system model that depicts the SFC Placement

problem. We defined the variables, decision variables, and restrictions. In the next section, we will present the proposed heuristic for solving the problem.

4.5 SPEED Heuristics

This section details how SPEED processes SFC requests to construct the SFC placement plan, which represents our distributed approach to solving the SFC Placement Problem. Figure 12 illustrates the complete set of activities involved in this process. We divide the problem into five key steps: i) Manager Zone Selection, ii) Segmentation, iii) Execution Zone Selection, iv) VNF Allocation in Compute Zones, and v) Network Path Construction.

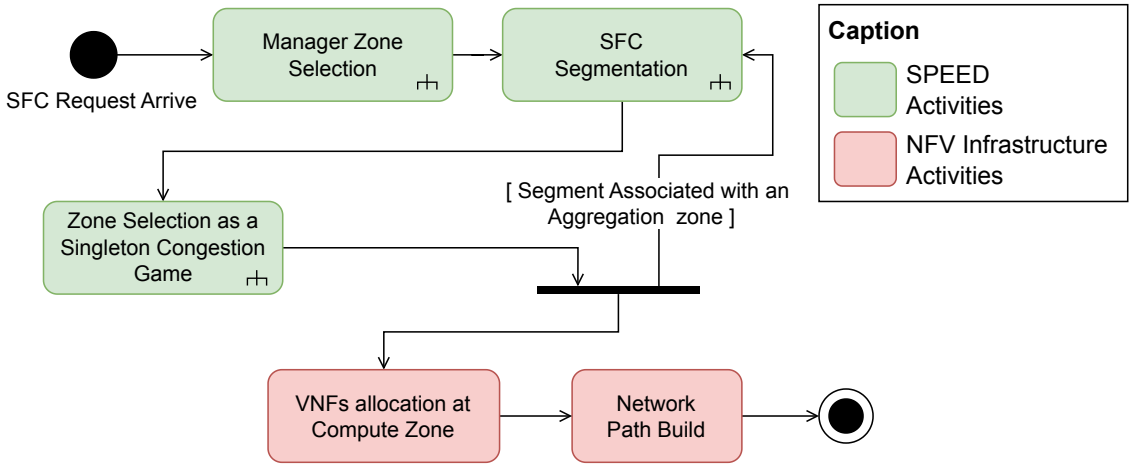


Figure 12: SPEED heuristic steps.

The **Manager Zone Selection** is the phase where the system selects the Aggregation zone that will coordinate the distributed SFC Placement. This selection considers the VNFs, the ingress node, and the egress node of the requested SFC. This process is straightforward to be done once the environment is modeled as a tree-like topology, as illustrated in Figure 8. This phase is detailed in Section 4.5.1.

The **SFC Segmentation** consists of creating segments based on i) the sequence of VNFs that compose an SFC and ii) the VNFs of an already created segment. This process allows the VNFs to be placed in the most suitable compute zone based on their requirements. Each segment will be placed into a different compute zone. This phase is detailed in Section 4.5.2.

In the **Zone Selection as a Singleton Congestion Game** phase, we select which zone will handle each segment created. We modeled this step as a Singleton Congestion Game. The Segmentation and the Execution Zone Selection will be executed multiple

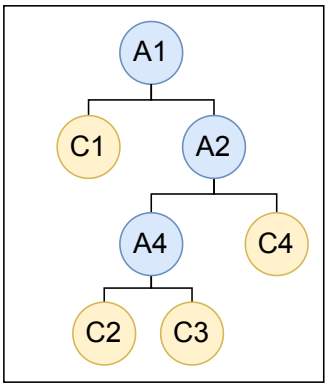
times until all the VNFs of the SFC request are placed. This phase is detailed in Section 4.5.3.

The **VNF allocation at Compute zone** is the VNF placement executed inside the selected Compute zone. The **Network Path Build** will create the network connectivity between all the VNFs of the service across multiple zones. These two phases are executed by the NFV Infrastructure.

4.5.1 Manager Zone Selection

The **Manager Zone Selection** is the process of selecting the Aggregation zone that coordinates the distributed SFC Placement. The Manager zone is (from the bottom to top in the topology) the most inferior Aggregation zone between the source and destination of the requested service. All the VNFs of the requested service must be placed in a computed zone underneath the selected zone. Thus, the selected zone should contain at least one child zone capable of executing each VNF requested in the service.

Figure 13 depicts an example of zone topology, and VNF Types availability in a hypothetical scenario composed of three Aggregation zones and three Compute zones. Suppose two services `r1` and `r2`. The VNFs of `r1` are `VNF1` and `VNF2`, and the VNFs of `r2` are `VNF1` and `VNF3`. Both services have the source in `C2` and destination in `C3`. The zone selected to manage `r1` is `A4` once `C2` and `C3` are children of `A4`, and they can execute all the requested VNFs. However, the zone selected to manage service `r2` must be `A1` once `C1` is the unique zone in the topology that can execute `VNF3`.



Available VNF Types per Zone

Zone	VNF Types
C1	VNF1, VNF2, VNF3
C2	VNF1, VNF2
C3	VNF1, VNF2
C4	VNF1, VNF2, VNF4

Caption

Compute zone

Aggregation zone

Figure 13: Example of Manager Zone Selection.

This process fails if the root zone cannot execute the VNFs of the requested service. We prioritize the selection of the lowest Aggregation zone in the tree-like topology to be the manager zone. This strategy tends to reduce the use of inter-domain links. Typically,

child zones of the same Aggregation zone will cooperatively share the infrastructure.

4.5.1.1 Distributed Destination Zone Search Algorithm

A key process during the zone selection is identifying the Aggregation Zone to which the destination compute zone `(dst)` belongs. Thus, in this section, we present the proposed algorithm to identify the corresponding Aggregation Zone associated with the SFC request. Figure 14 illustrates how SPEED performs a search for the destination zone within a hierarchical topology.

To improve the readability of the activity diagram, colors were assigned to activities based on their role in the process. *White activities* represent topology navigation searching for the compute zone. *Green activities* indicate the traversal of the success message when the destination zone is found. *Red activities* represent the traversal of the failure message when the destination zone is not present in the environment.

The proposed algorithm builds upon the distributed Depth-First Search (DFS) approach presented in (MAKKI et al., 1996). Our version was specifically designed to operate over a tree-like hierarchical topology. Therefore, the algorithm relies on the following assumptions: i) it assumes that the request initially arrives at the Aggregation Zone associated with the source zone `src`, ii) all Aggregation Zones have a parent zone, except for the root node, and iii) message exchanges occur only between parent and child zones.

The algorithm initiates a distributed asynchronous depth-first search over a hierarchical tree composed of aggregation and compute zones. When a search `R` arrives at an aggregation zone `AZ`, it first checks whether the destination compute zone `dst` is among the child compute zones of `AZ`. If not, `AZ` is appended to `R.rzc` preventing cyclic traversals and revisiting zones. The search then proceeds recursively through the child aggregation zones of `AZ`.

The child aggregation zones of `AZ` are filtered to remove those already visited, as recorded in `R.rzc`, and sorted by the number of compute zones they contain. If the list is not empty, each child is called asynchronously and in parallel to continue the search. Before starting its own search, each $zone_n$ checks whether the search was already completed in any zone listed in `R.rzc`. If `R.found = true`, the zone can terminate early, avoiding unnecessary processing.

If none of the child zones find the destination and there are no further zones to explore,

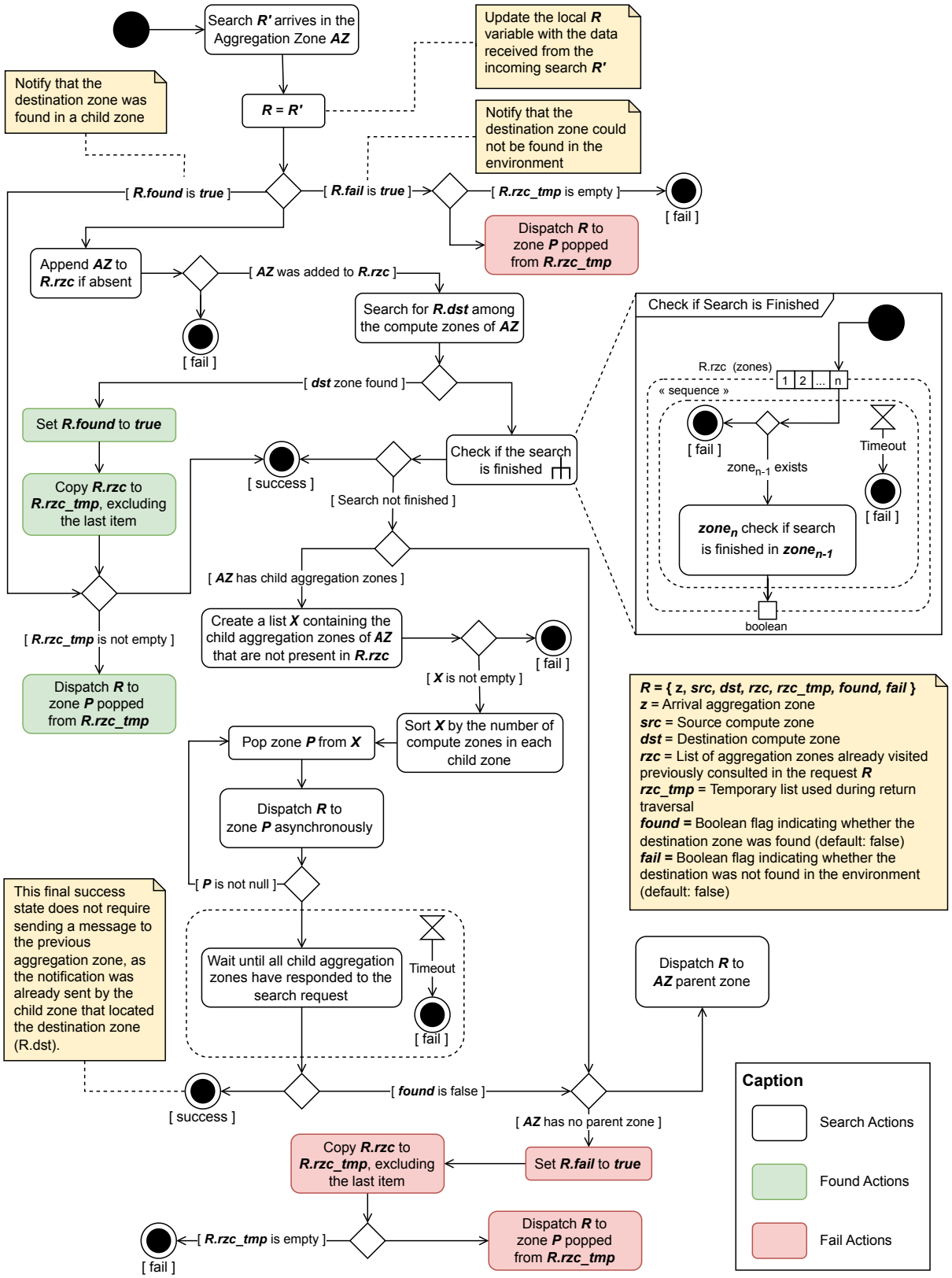


Figure 14: Distributed Depth-First Search.

the current zone forwards the request to its parent zone, continuing the depth-first search in the upward direction. Timeout mechanisms are in place to avoid indefinite blocking

caused by unresponsive zones. As the results propagate back through the visited zones, each one updates its local copy of `R`, ensuring consistency and enabling reliable distributed discovery of the target compute zone.

If all aggregation zones in the environment have been visited and the destination compute zone (`R.dst`) is not found, the search is considered a failure. In this case, the failure is detected at the root aggregation zone, which has no parent to escalate the request further. This root zone is responsible for triggering the failure message and initiating the return path.

The Aggregation Zone that initiated the search receives a message indicating whether the destination zone was found or not. Since this message is received asynchronously, the process that initiated the search must check for the result. This can be accomplished using strategies such as polling, message queues, callback URLs, or other event-driven mechanisms.

After the process described in this section, the path between `src` and `dst` in the tree-like topology is found, if it exists. With this information, it is possible to determine which aggregation zone should be used to coordinate the placement process. In the next section, we discuss how this selection is performed.

4.5.1.2 Finding the Manager Zone

After locating the destination compute zone in the environment, the next step is to determine which aggregation zone will act as the manager zone. To be selected as the manager zone, an aggregation zone must meet two conditions: i) both the source and destination compute zones must be its descendants, and ii) all VNF types required by the SFC must be available within its descendant zones. The manager zone is selected based on these requirements to ensure optimal coordination and resource availability.

Figure 15 illustrates the proposed algorithm for selecting the appropriate manager zone. Unlike the algorithm for locating the destination zone, presented in Section 4.5.1.1, this procedure requires a synchronous approach. Additionally, this process operates under the same communication constraints, wherein a child zone can only exchange messages with its parent zone.

To fulfill the first condition, we traverse the hierarchical topology using the list `R.rzc`, which is generated by the execution of the function that searches for the destination compute zone. The traversal proceeds upward through the tree until reaching a node

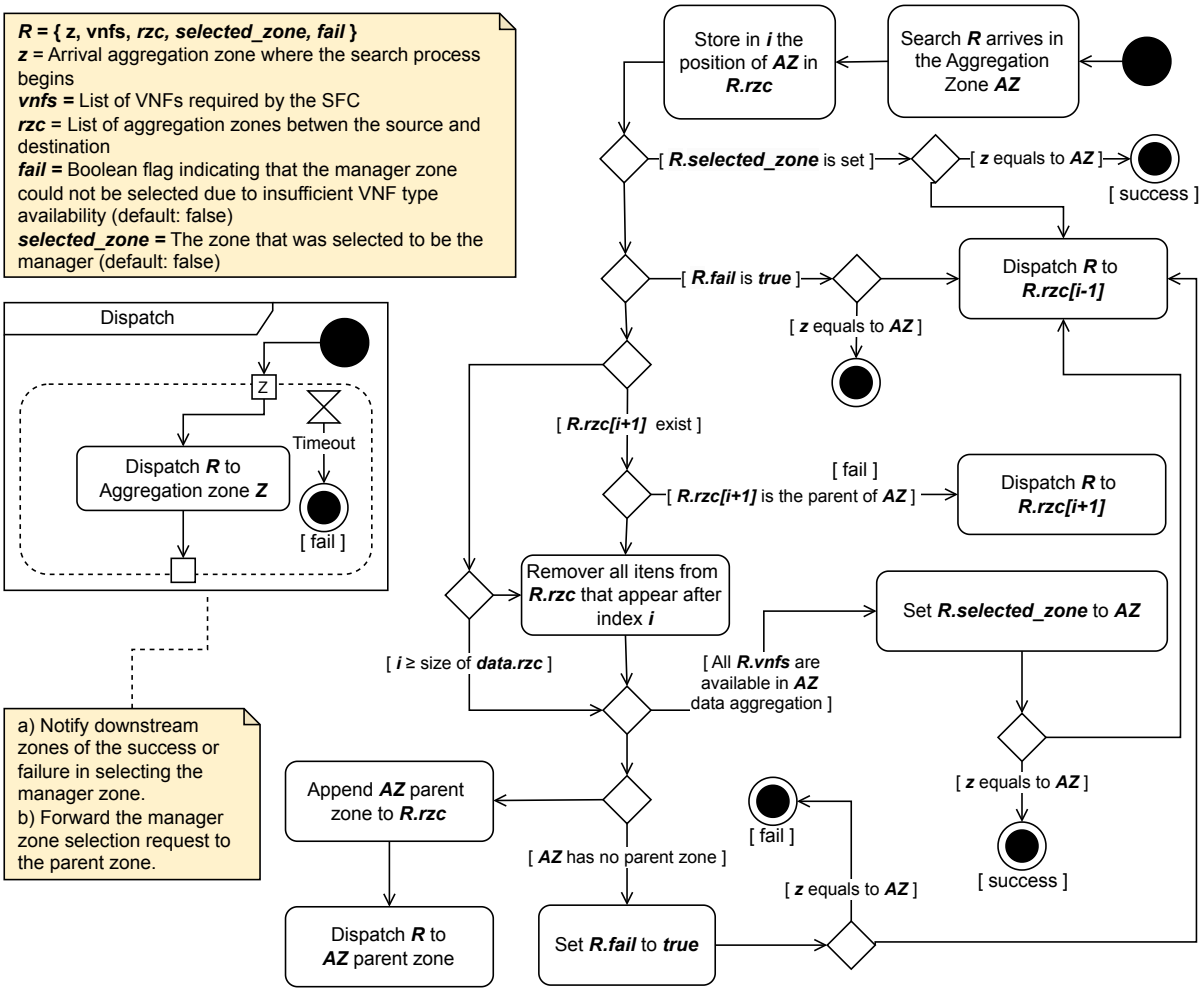


Figure 15: Activity diagram for selecting the manager zone.

where the next element in the list is no longer the parent of the previous one. At this point, the first condition is satisfied, as both the source and destination zones belong to the subtree rooted at this node.

After one node fulfills the first condition, we follow the process to fulfill the second condition. To satisfy this condition, we analyze the aggregated data of the current node. If all required VNF types are available, the node is selected as the coordination zone.

If some VNF types are missing, the selection request is forwarded to the parent node. The process continues upward until a suitable node is found or the root is reached. In the latter case, the SFC placement fails, indicating that the environment does not have all the VNF types necessary to satisfy the requested SFC.

In this section, we presented how the manager zone is selected based on the `src` and `dst` zones. The selected manager zone is responsible for initiating the placement process. In the next section, we present how the SFCs are segmented during the SFC placement

process.

4.5.2 SFC Segmentation

This section presents our approach to the SFC Segmentation problem in Edge-Cloud continuum. The segmentation process plays a crucial role in distributing VNFs across multiple domains while satisfying resource and latency constraints. In Section 4.5.2.1 we present the system model for the SFC Segmentation problem, and in Section 4.5.2.2 we describe the proposed heuristic that dynamically defines segments to improve placement feasibility and efficiency.

4.5.2.1 SFC Segmentation System Model

In this section, we describe the system model for our method for SFC segmentation in Edge-Cloud contiuum (BATTISTI; DELICATO et al., 2024). The model defines the environment and the elements where our proposed heuristic is executed to solve the SFC segmentation problem. Table 14 summarizes our symbols for our system model.

Table 14: SFC Segmentation System model symbols

Symbol	Description
Parameter Variables	
F	Set of VNFs in the SFC
S	Set of segments in the SFC
D	Set of domains
H_d	Set of nodes in domain d
$R_h = (c_h, m_h)$	Resource capacity of node h
c_h	CPU capacity of node h
m_h	Memory capacity of node h
cpu_f	CPU requirements for VNF f
mem_f	Memory requirements for VNF f
$\text{link}_{f,f'}$	Indicator of connectivity between VNFs
$\text{link\_cap}_{h,h'}$	Bandwidth capacity of the link between nodes h and h'
latency_s	Latency of segment s
latency_threshold	Maximum allowable latency for the SFC
Decision Variables	
$x_{f,s}$	Binary variable: 1 if VNF f is in segment s, 0 otherwise
$y_{s,d}$	Binary variable: 1 if segment s is placed in domain d, 0 otherwise

F is a set of VNFs. Each VNF represents a network function that is part of the SFC.

These functions need to be executed in a specific order to provide the desired network service. The cpu_f and mem_f represent the CPU and memory requirements for VNF f. This represents the resource load of a VNF, which must be accommodated within the available node resources.

D is a set of domains. Domains are administrative groupings of network resources that can execute VNFs. The challenge is to allocate VNFs across multiple domains while maintaining service requirements and improving resource use.

S is a set of segments resulting from the SFC segmentation. The segmentation process aims to divide the SFC into manageable parts that can be optimally distributed across domains.

H_d is a set of nodes in domain d. The nodes within a domain provide the computational resources needed to execute VNFs. The allocation of VNFs to nodes must consider resource constraints and the topology of the network. $R_h = (c_h, m_h)$ is the resource capacity of node h. Each node has a limited amount of resources, named CPU (c_h) and memory (m_h). The placement of VNFs must ensure that their total resource requirements do not exceed the capacity of the node.

The $\text{link}_{f,f'}$ is the connectivity indicator between VNFs f and f'. It ensures that VNFs within the same segment can communicate efficiently, following the required order of execution. The $\text{link\_cap}_{h,h'}$ defines the bandwidth capacity of the link. It limits the amount of data that can be transmitted between nodes, influencing the placement of VNFs to minimize internode communication delays. The latency_s defines the delay by processing VNFs within the segment. The latency_threshold acts as a constraint to ensure that the total latency across all segments does not exceed a specified limit of the SFC, thus maintaining the quality of service.

There are two decision variables in the system $x_{f,s}$ and $y_{s,d}$. The first, depicted in Equation (4.10), defines whether the VNF f is within the segment s. The second, depicted in Equation (4.11), indicates if the segment s is sent to domain d.

$$x_{f,s} = \begin{cases} 1, & \text{if VNF } f \text{ is included in segment } s \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

$$y_{s,d} = \begin{cases} 1, & \text{if segment } s \text{ is placed in domain } d \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

The problem is subject to some constraints. Equation (4.12) and Equation (4.13) ensure that, for each segment s placed in the domain d, the total resources required by the VNFs in the segment must not exceed the resources available in domain d.

$$\sum_{f \in F} x_{f,s} \cdot \text{cpu}_f \leq \sum_{h \in H_d} y_{s,d} \cdot c_h, \quad \forall s \in S, \forall d \in D \tag{4.12}$$

$$\sum_{f \in F} x_{f,s} \cdot \text{mem}_f \leq \sum_{h \in H_d} y_{s,d} \cdot m_h, \quad \forall s \in S, \forall d \in D \tag{4.13}$$

Equation (4.14) imposes the order preservation constraint of the VNFs. This constraint ensures that the sequence in which the VNFs are processed respects the predefined order in the SFC. This means that if function f_1 precedes function f_2 in the SFC, this order must be maintained across the segments. This constraint prohibits the creation of segmentation plans where the order of VNFs in the segments differs from their order in the SFC. For example, a plan like $p_1 = (\{v_1, v_3\}, \{v_2\})$ is not allowed for an SFC $(\{v_1, v_2, v_3\})$.

$$x_{f_1,s} \cdot x_{f_2,s} \leq x_{f_1,s} \cdot \sum_{s' \geq s} x_{f_2,s'}, \quad \forall f_1, f_2 \in F, f_1 \rightarrow f_2 \tag{4.14}$$

Equation (4.15) imposes the connectivity constraints. This constraint ensures that the connectivity requirements between VNFs within the same segment do not exceed the available link capacity between nodes.

$$\sum_{f,f' \in F} x_{f,s} \cdot x_{f',s} \cdot \text{link}_{f,f'} \leq \text{link\_cap}_{h,h'}, \forall h, h' \in H_d \tag{4.15}$$

Finally, Equation (4.16) defines the latency constraints. It ensures that the total latency of the segments does not exceed the maximum allowable latency of the SFC.

$$\sum_{s \in S} y_{s,d} \cdot \text{latency}_s \leq \text{latency\_threshold}, \quad \forall d \in D \tag{4.16}$$

The formulation presented in this section can be modeled as a Constraint Satisfaction Problem (CSP) (BRAILSFORD et al., 1999), where the variables represent VNFs, segments, and the constraints include resource limitations, latency requirements, and VNF ordering. The system model provides a comprehensive overview of the SFC

segmentation problem in multi-domain environments, defining the key elements and constraints necessary for the association of VNFs across different segments and domains. By incorporating resource, order preservation, connectivity, and latency constraints, the CSP formulation allows the problem to adapt to different objectives, such as optimizing placement success or minimizing deployment costs. In the next section, we present our method for solving the SFC segmentation problem.

4.5.2.2 Distributed Segmentation Strategy

This section presents our heuristic, named Distributed Segmentation Strategy (DSS), for solving the SFC segmentation problem. DSS is modular and decouples the segmentation and placement processes, making it adaptable to various SFC placement methods that operate in multi-domain environments. The proposed segmentation heuristic can be executed concurrently in multiple selected domains to execute the segment placement, encompassing the distributed nature of our approach.

Our proposed heuristic addresses the SFC segmentation problem by dynamically dividing SFCs into manageable segments for deployment in multi-domain environments. The method iterates through possible segmentation plans, selecting split points in the SFC to create segments that comply with the VNF order constraint. The heuristic generates valid segmentation plans and sorts them based on the placement strategy to achieve optimal SFC deployment.

The objective during the segmentation phase can vary depending on the requirements of the SFC placement method. For example, if the goal of SFC placement is to reduce the bandwidth consumption between domains, then the segmentation plan can be designed to concentrate the VNFs into a few domains. If the objective of the SFC placement is to maximize the placement success rate, the number of segments can be increased to use resources from different domains. The DSS approach can be configured to meet various goals according to the needs of the SFC placement strategy.

Algorithm 1 generates all possible split points for the VNFs of an SFC. The split point is the link that divides the SFC into two segments. Our proposed algorithm creates an array where each element is a boolean array that indicates in each element whether a link must be used as the split point or not. This boolean array is used to create the segments via Algorithm 2.

Algorithm 2 groups the VNFs (from an SFC or a segment) into distinct segments.

Algorithm 1: Creation of Link Splitting Map

```
Input: n, the number of VNFs
Result: linksToSegment, an array where each position is a split point plan
Initialization:
validPlans ← 2^(n−1)
arrayLength ← log2(n + 1)
linksToSegment ← Array(validPlans)
for i ← 0 to validPlans − 1 do
    binRep ← intToBinaryString(i)
    boolArray ← Array(arrayLength)
    j ← 0
    for each char in binRep do
        if char = '1' then
            boolArray_j ← True
        end
        else
            boolArray_j ← False
        end
        j ← j + 1
    end
    linksToSegment_i ← boolArray
end
return linksToSegment
```

This segmentation is based on a boolean array that specifies where the segments should be split. The goal is to create a segmentation plan that divides the VNFs into segments for further processing or deployment.

Figure 16 depicts the generation of all possible segmentation plans of an SFC with 3 VNFs using Algorithm 1 and Algorithm 2. As we can see in the figure, each possible segmentation plan indicates which links are used to split the segments. In the first segmentation plan, both link `L1` and `L2` are not chosen as split points, resulting in a segmentation plan with a single segment, $s_1 = \{v_0, v_1, v_2\}$ containing all the VNFs. In contrast, in the last segmentation plan, links `L1` and `L2` are selected as split points, producing 3 segments $s_1 = \{v_0\}, s_2 = \{v_1\}, s_3 = \{v_2\}$, with only one VNF.

Figure 17 depicts our proposed heuristic. The first parameter is a list of VNFs, which can represent either the entire set of VNFs of an SFC or a segment from a previously divided SFC. The second parameter defines the sorting strategy employed by the placement algorithm to achieve its objectives.

The first step consists of using Algorithm 1 to create the link splitting map. The number of VNFs requested is used as a parameter. The result of Algorithm 1 is an array

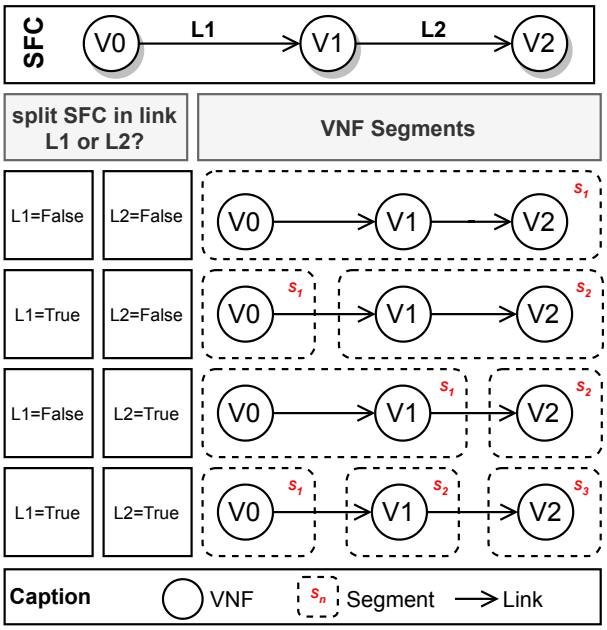


Figure 16: Example of segmentation with 3 VNFs.

with at least one element.

Algorithm 2: Creation of Segmentation Plans

Input: $vnfs$, array with the VNFs of the SFC, $linksToSegment$, boolean array indicating split points
Result: $segmentationPlan$, a list of segments
Initialization:
$segmentationPlan \leftarrow List()$
$i \leftarrow 0$
while $i < size(vnfs)$ **do**
  $segment \leftarrow List()$
  $segment.insert(vnfs_i)$
  $currentIndex \leftarrow i$
  for $j \leftarrow i$ **to** $size(linksToSegment) - 1$ **do**
    if $linksToSegment_j = True$ **then**
      break
    end
    $segment.insert(vnfs_{j+1})$
    $currentIndex \leftarrow j + 1$
  end
  $i \leftarrow currentIndex + 1$
  $segmentationPlan.insert(segment)$
end
return $segmentationPlan$

The next step consists of iterating over this array and, for each element, Algorithm 2 is executed. The result of Algorithm 2 is the segmentation plan based on the link splitting

map. All the possible segmentation plans are stored in list $\boldsymbol{S}$.

After creating all possible segmentation plans, they are sorted based on the sorting strategy provided by the placement algorithm. This sorting strategy is a method used to rank the segmentation plans created. After generating all valid segmentation plans, the sorting strategy sorts them based on specific criteria defined by the SFC placement method, such as minimizing resource usage, reducing latency, or optimizing placement success. This ensures that the most suitable segmentation plan, aligned with the objectives of the system, is tested first during the placement phase. Thus, the first element of the list is returned to the SFC placement component as the selected segmentation plan.

The subsequent step involves examining the newly created segments to determine if they have a candidate domain to be placed or if they need to be resegmented. Various strategies can be employed to verify whether the segment can be placed or not, and each placement component may approach this differently, such as using MIP, Game Theory, Auction, Full Mesh Aggregation, etc. If a created segment lacks a candidate domain capable of handling all its VNFs, the segment is reintroduced into the algorithm for resegmentation.

Segments with candidate domains for execution are handed over to the placement component. Once the segment reaches the chosen domain, the segmentation process can be performed once more, if necessary. This process persists until all requested VNFs are allocated to a domain capable of providing the necessary resources for their execution. Our proposed heuristic allows different segments to be segmented and placed in different domains simultaneously, reducing the overall time required for SFC placement.

In some cases, the resources available within the domains are insufficient to execute the SFC. The heuristics we propose stop when the segments created have a single VNF and there is no domain available to execute the VNF of the segment. In this case, the SFC placement fails.

The process of generating all segmentation plans involves creating subsets of VNFs, which corresponds to the "stars and bars" problem. For an SFC with n VNFs, the number of valid segmentation plans grows exponentially, almost $O(2^{(n-1)})$, where n represents the number of VNFs. This means that the time required to create all segmentation plans varies depending on the number of VNFs in the SFC.

The algorithm proposed in this section was developed to enhance our solution for the SFC Placement problem. However, it can also be applied to other contexts where a

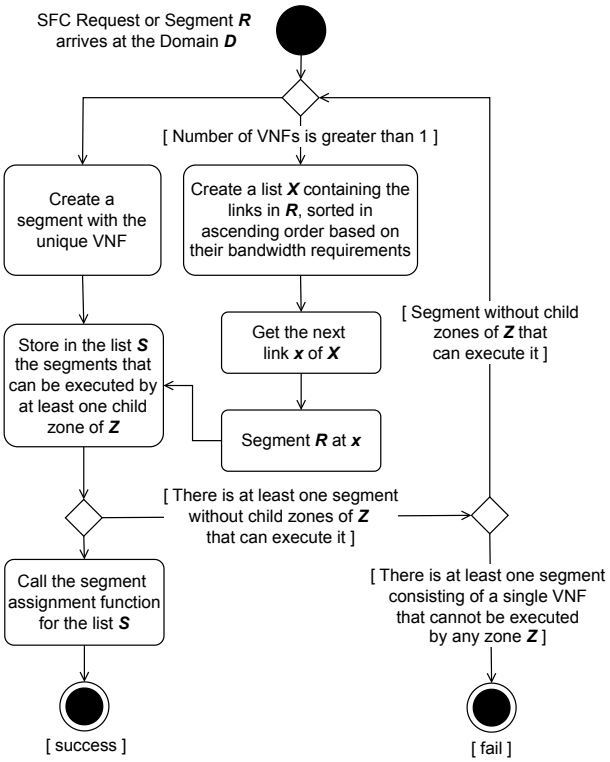


Figure 17: Segmentation Heuristic.

sorted set must be partitioned while preserving the original order within each partition. For example, ordered partitioning is also required in data streaming applications, where large data flows are divided into ordered chunks. In such cases, enumerating all possible partitions can assist in optimizing transmission strategies for latency, reliability, or load balancing. In the next section, we will present our heuristic for selecting the zone to execute a segment based on Singleton Congestion Game.

4.5.3 Zone Selection as a Singleton Congestion Game

In Section 2.6, 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 zone* 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.

Algorithm 3 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.

Figure 18 portrays an example of how our proposed approach can solve the SFCPP in a distributed fashion. The topology comprises 1 Access zone, named `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 4.2.2.

Algorithm 3: 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 $selZone$

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. 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 complete process to select the coordinating zone can be found in Section 4.5.1.

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.

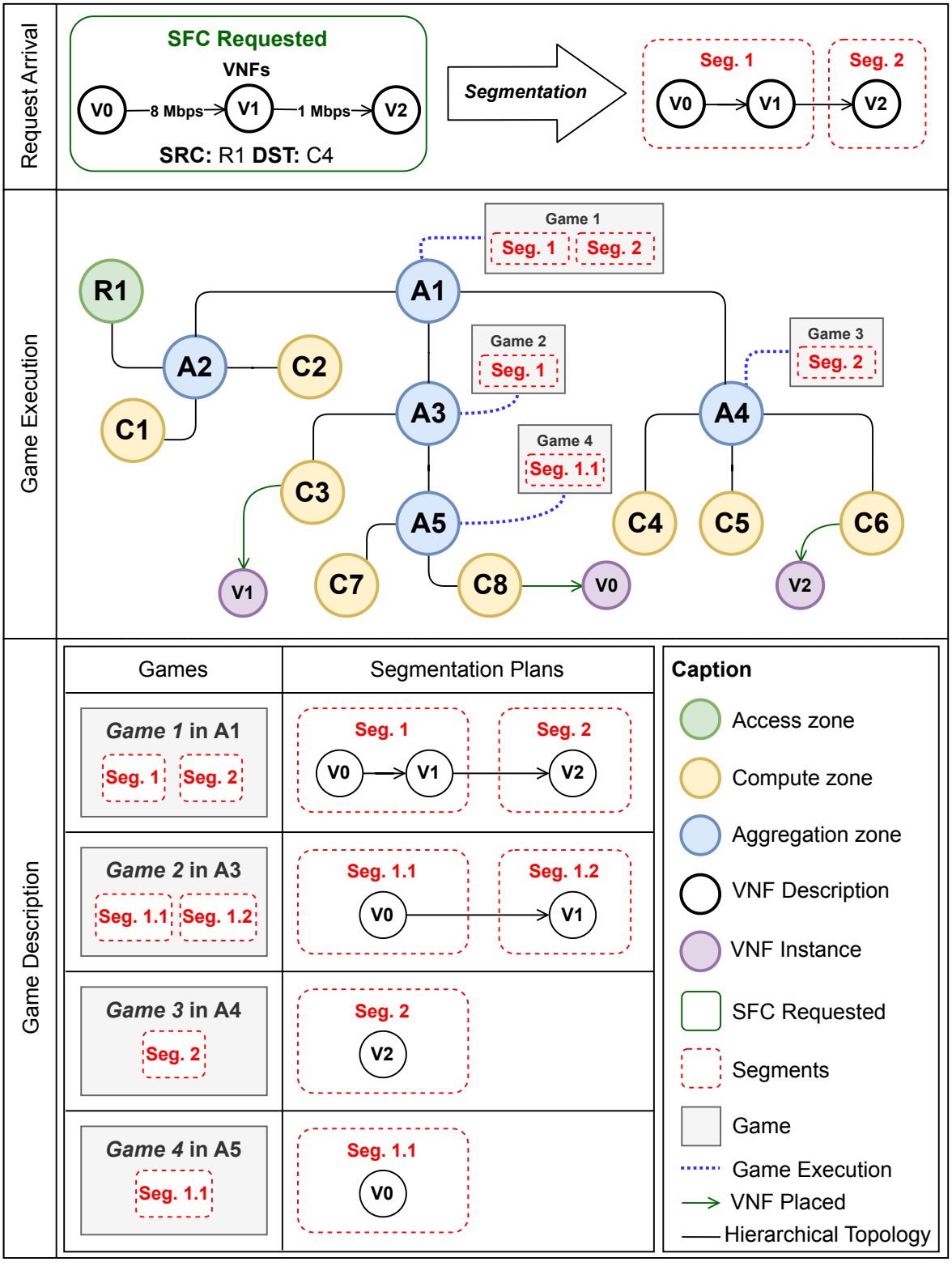


Figure 18: Running example of our approach solving the SFCPP.

4.5.3.1 Players Strategy Update

We modeled the SFC placement problem as an SGC, as described in Section 4.5.3. 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 (FUDENBERG et al., 1998). 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 (IWASE et al., 2017).

In our proposed solution, 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.

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 19 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 4.5.2.2, 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.

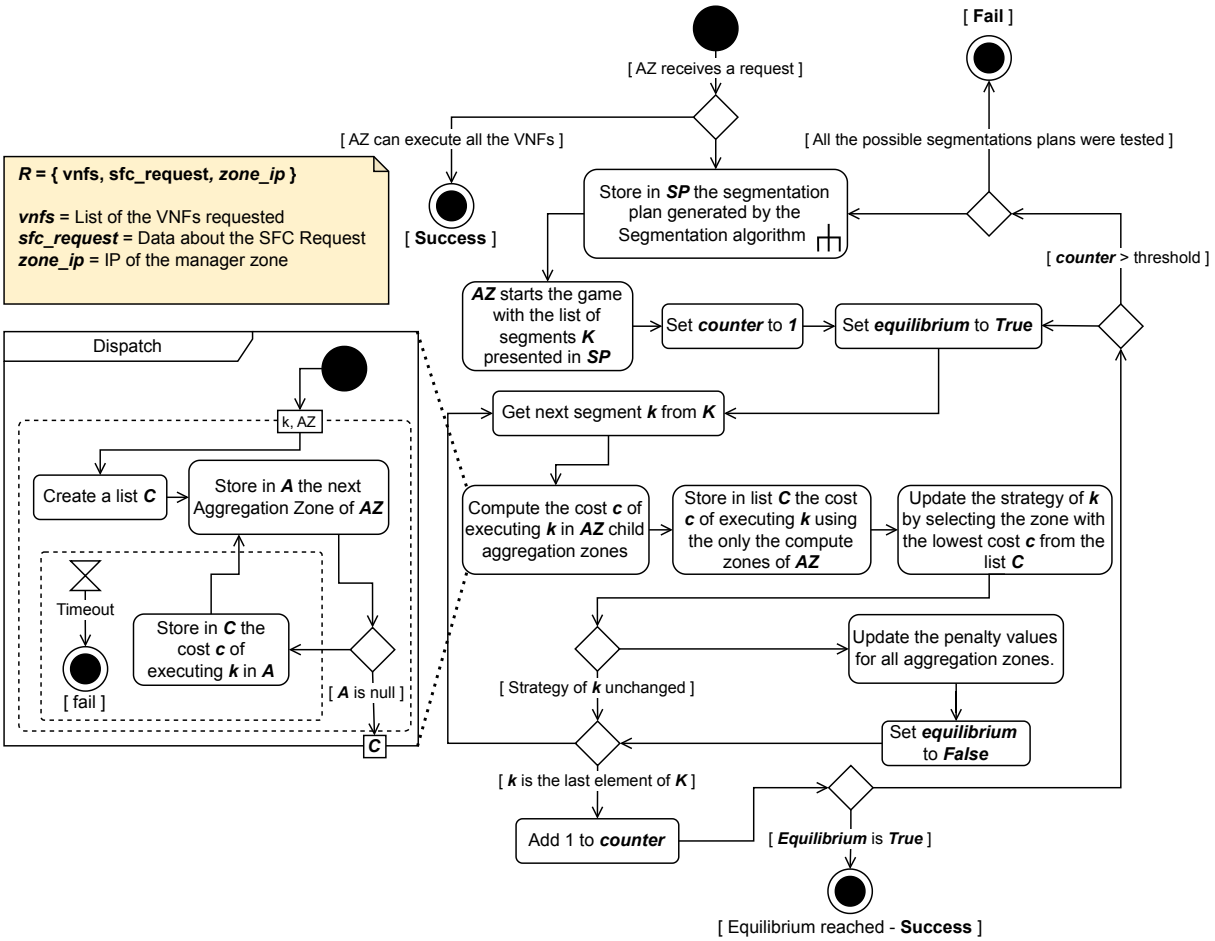


Figure 19: Players strategy update activity diagram.

Then, the current aggregation zone `AZ` initiates the game using the list `K` as input. The aggregation zone `AZ` is responsible for executing the game that determines the appropriate execution zone for each segment of the segmentation plan `k`. Each segment $\texttt{k}_\texttt{i}$ 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` ← `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 determined 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 correspondingly rises, 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 are suitable for environments with low to medium susceptibility to congestion.

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 4.17, 4.18 and 4.19 present how cost `c` is computed.

$$c(k_i, z) = \left(\sum_{f \in \mathcal{F}(k_i)} c_{\text{vnf}}(f, z) \right) \cdot penalty_z \tag{4.17}$$

$$penalty_z = \alpha \cdot e^{\beta n_z} \tag{4.18}$$

$$n_z = \sum_{k \in \mathcal{K}(z)} |\mathcal{F}(k)| \tag{4.19}$$

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.
- $\mathcal{F}(k_i)$ is the set of VNFs that compose segment k_i.
- $c_{\text{vnf}}(f, z)$ is the cost of execute the 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 number of VNFs associated with all segments allocated to zone z.
- $\mathcal{K}(z)$ is the set of 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, `equilibrium` $\leftarrow$ `False`. This indicates that another full round of strategy updates by the players is needed.

When a segment changes its assigned zone, the system must update 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 (i.e., k_i is the last element of K), 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 (i.e., `equilibrium` = `True`), 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 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. The next section details the operational aspects of SPEED.

4.6 Operation

This section presents essential aspects of the SPEED operation. The data aggregation process is presented in Section 4.6.1. We discuss how the SFC request arrives in the system in Section 4.6.2.

4.6.1 Data Aggregation

In our proposed approach, data aggregation is the process of combining and processing the data of the subordinated zones. The data aggregation allows each zone only to expose public information to other zones. The data aggregation strategy imposes constraints, but even with this strategy, SPEED is capable of executing the SFC placement in a distributed fashion.

The *Zone Manager* component runs in each Aggregation zone and periodically collects data about the subordinated zones. This data represents the VNFs that can be executed in the child zones. The data from all child zones are aggregated and informed to the parent zone. This data aggregation strategy is an important part of our proposed approach and allows SPEED to create the placement plan. Figure 20 depicts the sequence diagram to collect information about the child zones and send the aggregated data to the parent zone.

All Aggregation zones must execute the procedure to collect data about their child zones. This task can be executed periodically in a fixed interval, configured by the zone administrator, or every time a child zone reports some change in the VNF availability. The data aggregated is stored in the Data Aggregated Storage in the SPEED component.

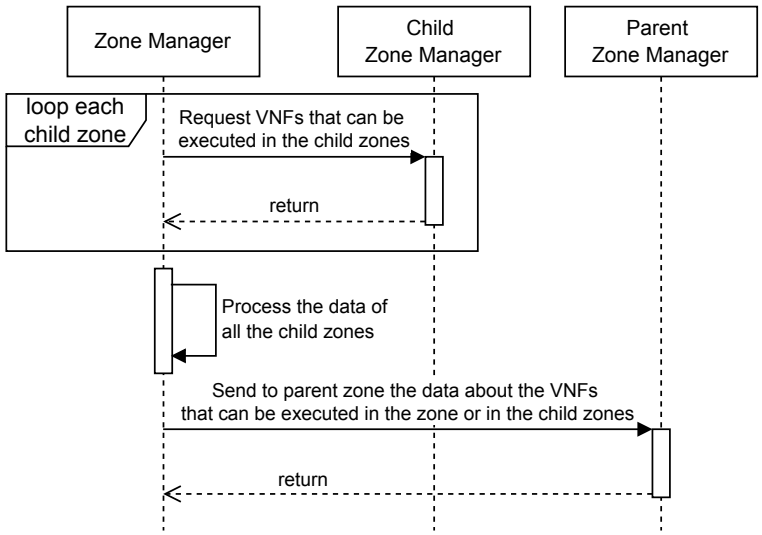


Figure 20: Aggregate data from child zones diagram.

4.6.2 SFC Request Arrival

The main goal of the proposed approach is to allocate an SFC in a multi-domain environment in a distributed fashion. Figure 21 depicts the actions and the main components involved during the placement of an SFC. Firstly, the User requests the SFC via the Client Portal. The Request arrives in the *Orchestrator* component of the Access zone where the User is associated to. Then, the *Zone Manager* component is responsible for finding the manager zone κ where the source and destination of the requested SFC are subordinated. Thus, zone κ will be responsible for coordinating the distributed placement of the SFC request.

Zone κ will create the segments based on the Algorithm 3. After that, each segment will be delivered to the selected zone, and the zone will execute the placement of the VNF in its Compute zone. After the placement of each VNF, zone κ is informed by the computed zones.

4.7 Conclusion

In this chapter, we introduced our proposed approach, named SPEED, to solve the SFC Placement Problem in a distributed environment using Game Theory. We first defined the environment and the entities involved in the SFC placement process. Then, we presented the system model that describes the structure and constraints of the placement problem.

We also described the architecture required to share the data necessary for building

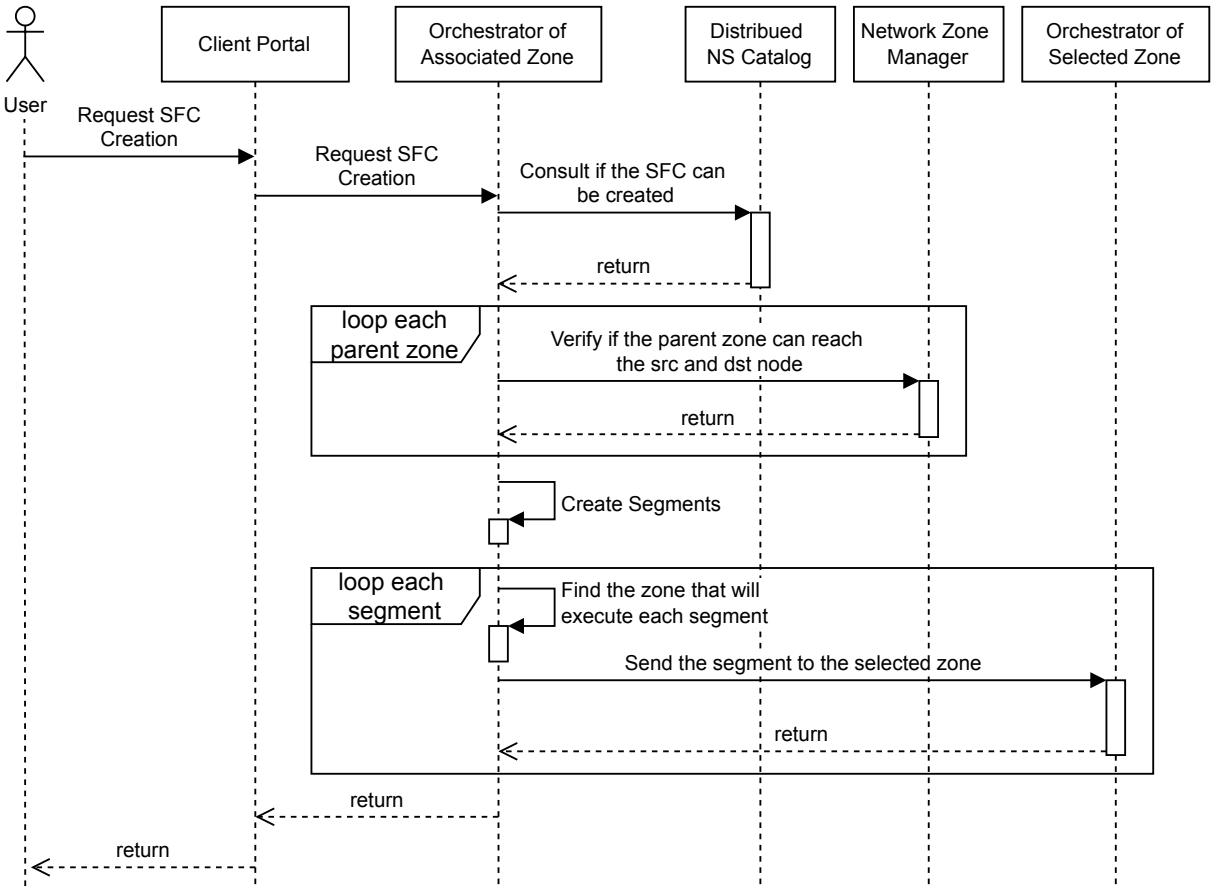


Figure 21: SFC Request diagram.

the placement plan. In addition, we explained the algorithms used to select the zone manager and to segment the SFC. Finally, we showed how the Singleton Congestion Game is played to determine which zone will execute each segment.

In the next chapter, we present the execution of the proposed algorithm in a simulated environment. The experimental results related to SFC segmentation are discussed in detail. Additionally, the creation and performance of the SFC placement plan are evaluated using multiple strategies.

5 Performance Evaluation in a Simulated Environment

In this chapter, we describe the execution of our proposed solution in a simulated environment. In Section 5.1 we present the experiments conducted for evaluating the performance of our proposed SFC Segmentation algorithm against other strategies. In Section 5.2 we present the experiment which evaluates our proposed method for the creation of the SFC Placement Plan and compares it to other methods.

5.1 SFC Segmentation

This section presents the results of our performance evaluation for our proposed solution for solving the SFC segmentation in different SFC placement scenarios (BATTISTI; DELICATO et al., 2024). The experiments were conducted on a system with an Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz. The system is equipped with 4 physical cores (8 threads) and a cache size of 8MB. The total available memory is 20GB. Our proposed heuristic solves the SFC segmentation problem in the Edge-Cloud continuum and can be used with different SFC placement methods. This unique characteristic of our proposed heuristic imposes challenges for a fair comparison with the solutions already proposed for the SFC segmentation problem. Therefore, we conduct a series of experiments that evaluate the time necessary to build the segmentation plan and the success rate of the SFC placement with different SFC placement configurations.

To evaluate the proposed solution, we conducted a series of experiments. In these experiments, the **independent variables** are i) the *Number of VNFs*, which indicates how many VNFs are in the SFC, and ii) the *Number of SFC Requests*, referring to the number of SFCs that were requested by the users. The **dependent variables** are i) the *Segmentation time*, representing how long the process took to create the segmentation plans, and ii) the *Service Placed*, which measures the percentage of SFCs placed based on the segmentation plan.

Table 15: 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
Bandwidth capacity in inter-domain link	1000 Mbps
Bandwidth capacity in intra-domain link	2000 Mbps
Link Delay	[2, 5] ms
CPU Cost	[0.001, 0.005] $/s
Memory Cost	[0.001, 0.004] $/GBs
SFC Size (Number of VNFs)	[3 - 6]
CPU demand of VNF	[1,4] GHz
Memory demand of VNF	[1,5] GB
Traffic rate requirement	[100, 500] kbps
Traffic routing cost parameter	[0.02, 0.05] $/Mb
Maximum tolerated delay	[0, 50] ms

Table 15 shows the simulation parameters, which were defined on the basis of (CHEN, C. et al., 2022). We use Internodes backbone topology, from the Internet Topology Zoo (KNIGHT et al., 2011). Each of the 64 nodes in the topology was considered a domain. We create scenarios with 5, 10, 20, 50, and 100 SFC requests. The SFC requests arrive in an interval following a Poisson distribution. The VNF Types in the SFC represent network services such as firewalls, intrusion detection, cache, flow monitor, and load balancer. The minimum CPU and memory requirements for each VNF Type were obtained from the respective developer websites. The simulation was implemented using SimPy (SIMPY DEVELOPMENT TEAM, 2025). The simulations were executed 10 times for each scenario. The confidence level of the results is 95%. We adopt the SFC with 3 - 6 VNFs, equivalent to the value presented in (LI, Y. et al., 2023).

Solving the SFC Placement Problem takes considerable time (MACEDO et al., 2023). Therefore, the segmentation process must be time efficient to ensure that it does not compromise the overall duration of SFC placement. Figure 22 illustrates the average time required to create the segmentation plan as the number of VNFs in the SFC changes. The x-axis indicates the number of VNFs in the SFC, while the y-axis represents the average time it takes to generate the segmentation plan, and the y-axis is in logarithmic scale. We compared the approach with a naive method that generates all possible combinations of VNFs. The results show that by focusing exclusively on the generation of valid plans, the DSS approach reduces the overall generation time.

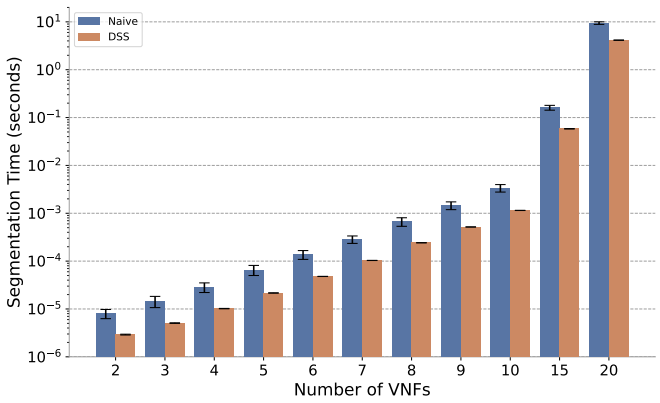


Figure 22: Segmentation Time by SFC Size.

Figure 23 shows the success rates of SFC placement using a placement method based on Singleton Congestion Game (ROSENTHAL, 1973), which is a type of game in Game Theory where each player selects a single resource from a set, and the cost or utility for using that resource depends on how many players choose the same resource; thus, players aim to minimize their individual costs, which increase as more players select the same resource, creating "congestion" on that resource. The implementation was configured using our proposed heuristic with two segmentation strategies i) Fixed Segment Size, where all created segments contain the same number of VNFs; and ii) Variable Segment Size, where the number of VNFs in each segment varies depending on characteristics of the environment. The x-axis represents the number of SFC requests [5, 10, 20, 50, 100], while the y-axis shows the percentage of SFC placed successfully. The results indicate that the variable segment size strategy achieves higher placement success rates, especially as the number of requests increases. This suggests that allowing segment sizes to adapt dynamically to changing conditions leads to more efficient resource allocation and improved network performance compared to a fixed segment size approach.

Figure 24 presents the execution of DSS in an SFC placement method based on a greedy approach. The Placement was executed using two segmentation strategies, the first one tries to reduce the placement cost selecting the nodes with lower execution cost and the second 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.

Based on the results presented, we can conclude that our proposed heuristic achieved its objective. Firstly, as shown in Figure 22, the time required to generate the placement

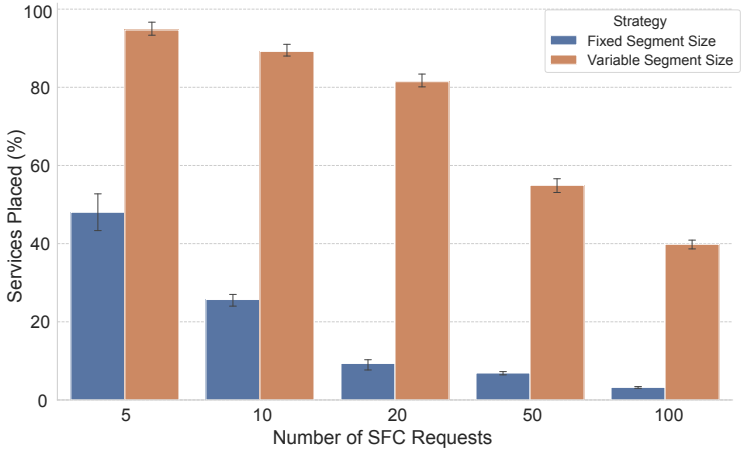


Figure 23: Impact of Segment Size on SFC Placement Success Rates.

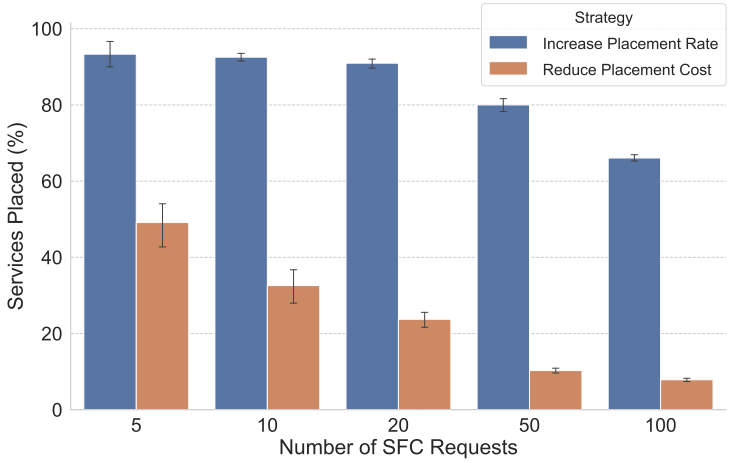


Figure 24: Impact of Cost Minimization on SFC Placement Success Rates.

plan for an SFC with 20 VNFs is less than 10 ms, thus not negatively impacting the total SFC placement time. Secondly, we integrate DSS into two distinct SFC placement methods. The results in Figure 23 come from a Game Theory approach, while the results in Figure 24 are based on a Greedy approach. This shows that our approach can be combined with multiple SFC placement methods. Finally, the results in Figure 23 and Figure 24 demonstrate that our approach can serve multiple purposes, including reducing SFC placement costs and increasing the SFC placement rate.

5.2 SFC Placement Plan Creation

In this section, we present a comparison between SPEED and other approaches. The evaluation is based on two key metrics: i) *SFC Placement Success Rate*, defined as the ratio of successfully placed SFCs to the total number of requested SFCs, and ii)

Average Execution Cost, which represents the monetary cost associated with executing the requested SFCs. Due to the high complexity of the problem, the ILP method could only be included in the first experiment. The compared approaches are:

- **Single Domain** is a decentralized approach that selects the resources to execute the SFC in the same domain where the request arrives;
- **Auction** is a distributed approach that selects the domain where the SFC will be executed using an auction strategy based on (LIU et al., 2020) and (AVASALCAI et al., 2019);
- **Integer Linear Programming (ILP)** is a centralized method that generates the best response, based on the system model defined in Section 4.4;
- **SPEED**, our proposed approach, described in detail in Chapter 4.

Table 16 shows the simulation parameters, defined on the basis of (CHEN, C. et al., 2022). We use Internodes backbone topology, depicted in Figure 25, from the Internet Topology Zoo (KNIGHT et al., 2011). We organized the Internodes topology into 10 different tree topologies based on the node centrality. Figure 26 depicts a fragment of how the zones were organized. Each node in the topology was considered a domain. We created scenarios with 5, 20, 50, 80, and 100 SFC requests. The requests arrive in an interval following a Poisson distribution. The VNF Types in the SFC represent network services like firewalls, intruder detection, cache, flow monitor, load balancer, and Wan-opt. The CPU and memory requirements for each VNF Type were compiled from the vendor's website for each VNF Type. The simulations were executed 10 times for each scenario. The confidence level of the results is 95%.

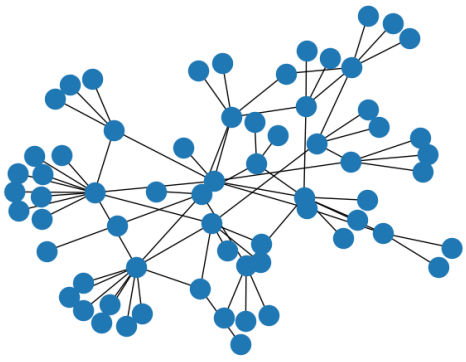


Figure 25: Internode Topology.

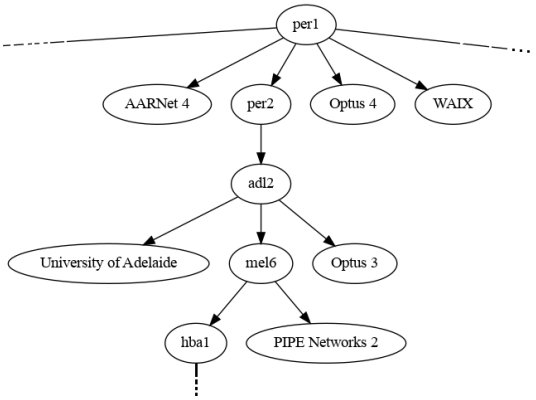


Figure 26: Example of the topology used in the experiments.

Table 16: 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
Bandwidth capacity in inter-domain link	1000 Mbps
Bandwidth capacity in intra-domain link	2000 Mbps
Link Delay	[2, 5] ms
Description	**Values**
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 parameter	[0.02, 0.05] $/Mb
Maximum tolerated delay	[0, 50] ms

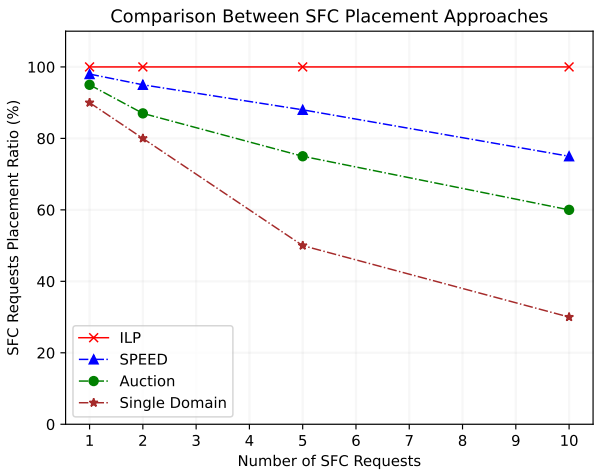


Figure 27: SFC Placement Success Rate.

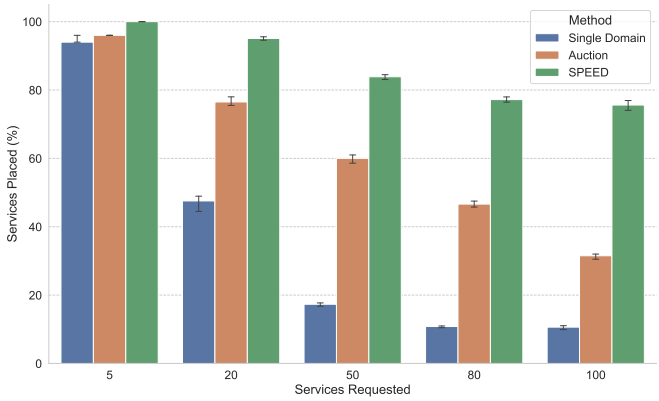


Figure 28: SFC Placement Success Rate.

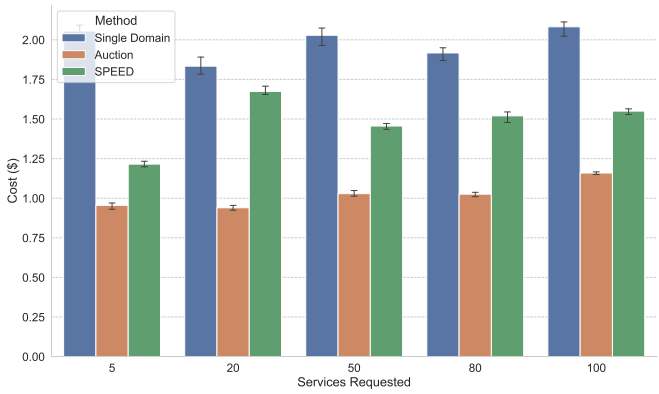


Figure 29: SFC Average Execution Cost.

Figure 27 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 20% lower success rate than the ILP when handling 10 requests. The SPEED method, despite its lower success rate in this context, brings benefits such as enhanced scalability and fault tolerance that are often constrained in centralized solutions like ILP. Therefore, while the ILP method shows superior performance in controlled small-scale scenarios, the practical applicability of the SPEED approach in larger and complex networks will be present in the next experiments.

Figure 28 compares 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

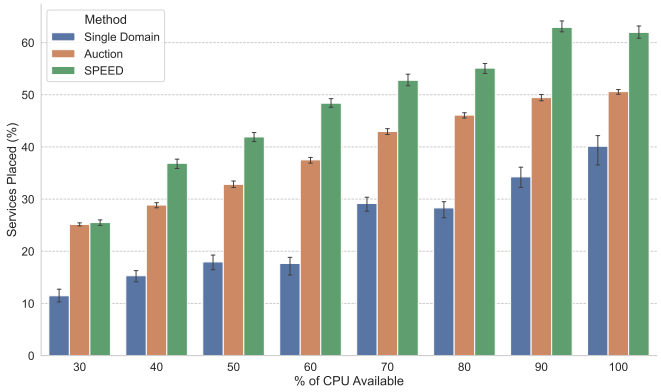


Figure 30: SFC Placement Success Rate x CPU Available.

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 29 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 the other approaches. Therefore, our proposed approach, despite increasing the number of requests placed, maintains similar costs in the execution of each SFC. Thus, adopting our approach, the service providers will lease more resources with similar prices to the users and will also increase their profit.

In certain environments, the resources of a zone were not exclusively reserved for the execution of SFCs. Figure 30 illustrates a scenario in which the number of SFC requests was set to 50, while the resources available on the nodes to execute SFCs were restricted to a percentage of its total resources. The results show that SPEED outperforms the other approaches in scenarios with scarce available resources.

5.3 Conclusion

In this chapter, we evaluated the performance of our proposed solution. We formulated a series of experiments in a simulated environment created using the SymPy library. The results demonstrated that our heuristic efficiently segments SFCs and improves placement success rates while maintaining low execution costs. These findings confirm the effectiveness and flexibility of our approach when compared to existing solutions in

multi-domain scenarios.

In the next chapter, we will evaluate our proposed solution in an execution environment built using real components. The compute and network infrastructure is provided by Kubernetes clusters. The main components were implemented according to the specifications described in Chapter 4.

6 Performance Evaluation in a Real Environment

In this chapter we evaluate the efficiency of the proposed solution, SPEED, in real distributed environments. Our goal is to demonstrate how the architectural choices and algorithms developed, including hierarchical metadata aggregation, manager zone selection, and distributed SFC placement, contribute to increasing the SFC Placement rate, ensuring the feasibility of operating in large-scale, multi-domain Edge-Cloud continuum. Through running examples and experiments, we aim to validate the core design principles of SPEED and compare its performance against baseline strategies under different operational conditions.

This chapter is organized as follows: in Section 6.1, we present the elements that comprise the compute and network infrastructure, which represent the physical environment in which our solution components are deployed. In Section 6.2, we present the implementation of the components previously outlined in Section 4.3. Finally, in Section 6.3, we describe the experiments executed in this real environment.

6.1 Environment Setup

In this section, we describe the environment used to deploy the SPEED Components and the SFCs. We selected the Kind tool[1] for building the multi-domain environment. We chose the SPIRE platform to establish a hierarchical topology by creating federations between parent and child domains. To execute the VNFs within the SFC and bind them together, we utilize Network Service Mesh (NSM). We will describe each technology in detail in the following sections.

[1] https://kind.sigs.K8s.io/

6.1.1 Compute and Network Environment

Our proposed approach, SPEED, is designed to run in a distributed environment composed of multiple domains. In our experimental setup, each domain corresponds to a Kubernetes (K8s) cluster, with each cluster consisting of multiple compute nodes. To ensure environmental heterogeneity, the number of compute nodes in each domain and the resources available in each compute node vary across domains.

Each K8s cluster is deployed using Kind[2], a tool for running K8s clusters within Docker container-based nodes. Originally designed for testing Kubernetes itself, Kind is also useful for local development and test environments. In our experimental setup, the environment bootstrap was automated using shell scripts and configuration files to create and configure the K8s clusters. The source code of our implementation and the scripts to execute the experiments can be found at https://github.com/anselmobattisti/speed_api.

Using the Kind tool, we can specify the resources available for each node within a K8s cluster. This flexibility allows creating a wide range of scenarios, enabling an extensive evaluation of SPEED in different contexts. For instance, we can configure domains at higher levels of the hierarchy to have more resources than those at lower levels, simulating a three-tier architecture where cloud nodes have more resources than edge nodes.

Our proposed solution operates in an environment with a hierarchical topology. In this topology, each domain is associated with a parent domain and may have zero or more child domains. To establish this structure, we define the relationships of the domains in a configuration file that is used during the environment bootstrap. The relationship between domains is enforced through the SPIRE platform.

SPIRE is an implementation of the SPIFFE API[3] that performs node and workload attestation to securely issue SPIFFE Verifiable Identity Documents (SVIDs). In our context, the workload refers to the network traffic transferred between the VNFs. Verifying the SVIDs attached to workloads and nodes is possible to check the authenticity of the entities based on predefined conditions. Each domain has its own SPIRE server installation that is responsible for authenticating the incoming workloads.

We identify a workload as authorized only if it originates from the parent domain. Consequently, workloads sent from the child domain to the parent domain, as well as those received from external domains outside the federation, are rejected. This approach

[2] https://kind.sigs.K8s.io/
[3] https://spiffe.io/

increases the security of a dynamically hierarchical multi-domain environment.

6.1.2 SFC Execution

Executing VNFs and SFCs in a real distributed environment presents several challenges, including heterogeneity in compute and network resources, coordination of VNF deployment and management, and the security and control of network traffic (SANTOS et al., 2022). These challenges must be addressed to ensure efficient operation and performance in a distributed environment. Therefore, we need to carefully select the appropriate tools to create a suitable testing environment that is capable of providing the characteristics required to test our proposed approach.

To execute the VNFs of the SFC, we use the Network Service Mesh (NSM) platform. NSM is a hybrid, multi-cloud IP service mesh that facilitates workload execution across heterogeneous environments through a set of APIs (DAB et al., 2020). It enables the integration of diverse compute environments, including containers, pods, virtual machines, and bare metal, to process workloads. Figure 31 depicts an environment where the components of a Network Service (NS) are executed in different domains.

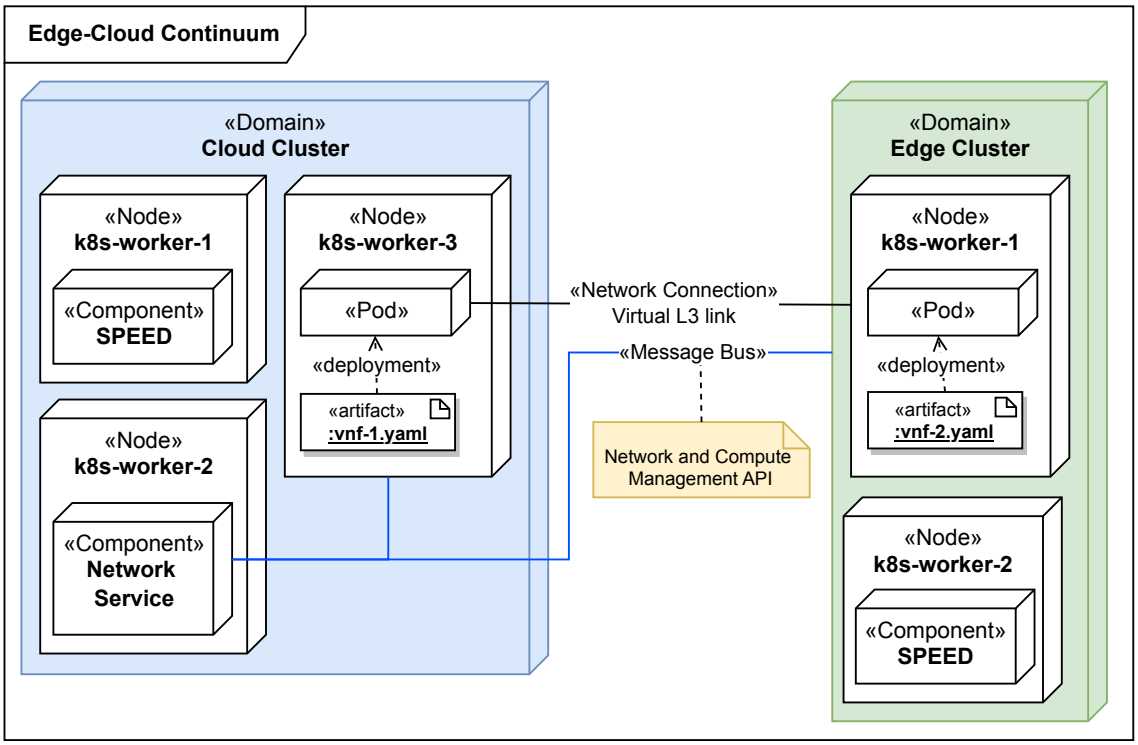


Figure 31: Network Service Deployment Diagram.

In the NSM architecture, SFCs can be defined as Network Service Compositions (NSCs). An NSC consists of multiple processing entities that enable complex workload

execution. The requested payload traverses each entity in the NSC, being processed before being sent to the next entity in the chain. In our experimental setup, these entities are the VNFs that compose the SFC.

Figure 31 illustrates how NSM utilizes the available compute infrastructure to deploy the entities for executing each part of the NSC. In the presented case, where the compute environments are K8s clusters, each NSC entity runs inside a K8s pod. However, if the compute environment was based on a traditional virtualization engine such as VMware, KVM, or others, the resulting environment would consist of virtual machines (VMs) instead.

NSM leverages Layer 3 (L3) networking to connect the entities of a Network Service, enabling direct point-to-point IP communication. Instead of using overlay networks, NSM establishes L3 tunnels using technologies such as VXLAN, SRv6, or GRE between the Network Service entities. This approach simplifies communication and enables dynamic service composition across multi-domain environments, allowing VNFs to handle a wider range of payloads, including multimedia flows that typically operate at L3 rather than L7, which is the conventional approach in traditional service mesh architectures (SONG et al., 2024).

Figure 31 also illustrates how NSM utilizes the available network infrastructure to establish connections between NSC components. In this example, NSM creates a network interface inside each pod, connecting it to a dedicated network that forwards network service packets. This network is implemented using a set of NSM components, including VL3 IPAM, which manages the IP addresses of the interfaces that encompass the virtual links between the network service entities.

In this section, we presented the environment in which SPEED is executed. We used the kind tool to create multiple K8s clusters, representing the network and compute environment. We also defined the hierarchical topology using the SPIRE platform. Additionally, we introduced NSM, the engine responsible for executing the VNFs of the SFC. In the next section, we will describe how SPEED utilizes this infrastructure to perform SFC placement in a distributed manner.

6.2 Implementation

In this section, we present how we implemented the core components of our proposed solution. We use the concept of Custom Resources in the K8s environment to store data

about the available VNF Types in each domain. The data retrieval process is performed using a VNF Type Operator that interacts with the K8s API to collect the data about each node. We also create a VNF Type Data Aggregation API to enable the communication from child to parent domain to perform the data aggregation process. Figure 32 depicts the main activities performed by the components to execute the data aggregation process over the distributed infrastructure.

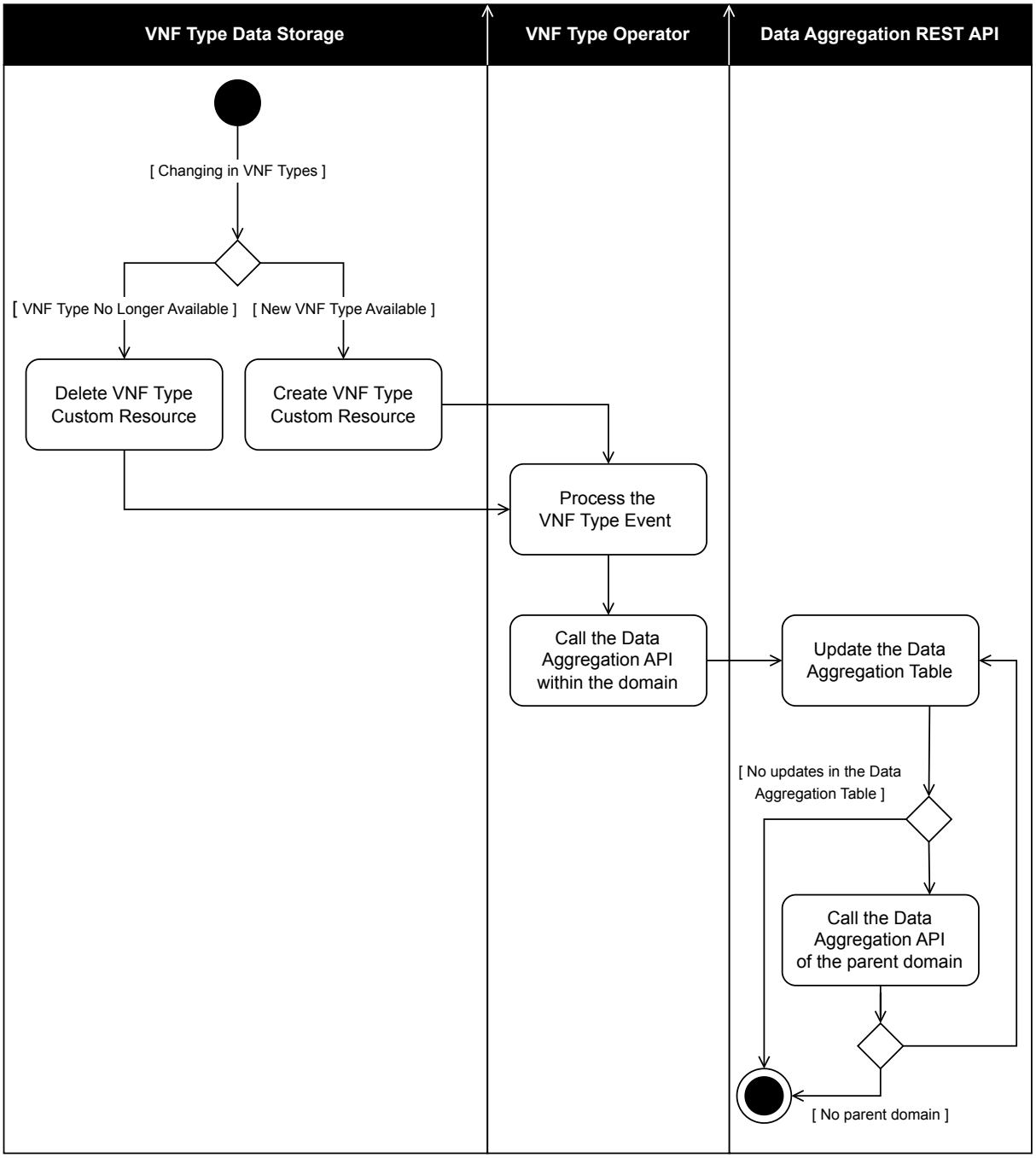


Figure 32: Activity Diagram of Data Aggregation Process.

6.2.1 VNF Type Data Storage

A key challenge in SFC placement within a distributed environment is storing data about the VNF Types available in each domain. Since our environment consists of multiple K8s clusters, we use K8s native features to handle the VNF Types. Specifically, we use Custom Resource Definitions (CRDs) to extend K8s native functionalities to store the required VNF Type data.

CRDs extend K8s by allowing users to define and manage Custom Resources (CR) tailored to specific applications (DOCUMENTATION, 2024). In our case, we create a CRD that specifies the necessary data about VNF Types required for SFC placement. During the bootstrap phase, we install the CRD responsible for managing VNF Type data in each domain.

When a change occurs in the VNF Types within the environment, it results in modifications to the corresponding VNF Type CR. If a new VNF Type becomes available, a corresponding CR is created based on the VNF Type CRD. On the other hand, if a VNF Type is no longer available, the associated CR is deleted. In Figure 32, component *VNF Type Data Storage* illustrates the creation and deletion of VNF Type CRs, which trigger events that are processed by the VNF Type Operator component. By leveraging the Kubernetes CRD feature, we can manage VNF Types as native K8s objects, similar to pods, deployments, and other standard resources.

The key advantage of making VNF Types managed as K8s objects is that they become part of K8s reconciliation loop (BREITGAND et al., 2021). As a result, K8s continuously monitors the lifecycle of these objects and generates events about their state. This allows for intercepting these events to enable proactive decision-making and the execution of complex tasks.

Operators are specialized controllers that automate the deployment, management, and lifecycle of complex applications and resources in a Kubernetes environment[4]. They use CRDs to define and control application-specific logic, enabling tasks such as deployment, scaling, and healing. By continuously monitoring the cluster state, Operators ensure that applications run as expected with minimal manual intervention.

The VNF Type Operator was created using the Kubernetes Operators Framework (Kopf)[5]. We execute one VNF Type Operator inside each domain of our distributed

[4]https://kubernetes.io/docs/concepts/extend-kubernetes/operator/
[5]https://kopf.readthedocs.io/en/stable/

environment as a POD. Figure 32 also presents the VNF Type Operator component that processes the event generated by the VNF Type CRs calling the VNF Type Data Aggregation API. This step will be described in detail in the next section.

6.2.2 Data Aggregation

Data aggregation is a fundamental process of our proposed approach, as discussed in detail in Section 4.2.2. In summary, our solution addresses the challenge of storing and managing VNF Type data in a large-scale Edge-Cloud continuum by implementing a hierarchical data aggregation strategy. Rather than maintaining a global view of the entire topology, each Aggregation zone receives and processes data from its child zones, ensuring scalability and efficiency. Combined with other strategies adopted in our proposed approach, we can improve SFC placement by enabling localized decision-making while preserving data privacy and minimizing overhead in a distributed multi-domain infrastructure.

The first step in the data aggregation process is storing, within each domain, information about which VNF Types can be executed on each compute node of such domain. This is accomplished using Kubernetes CRDs, as described in Section 6.2.1. Figure 32 shows the *Data Aggregation REST API* component which stores the data processed by the VNF Type Operator. At this stage, the request is performed directly to the Data Aggregation API instance running within the same domain as the VNF Type Operator.

During the bootstrap phase, we deploy a REST Data Aggregation API component in each cluster to receive data from the VNF Type Operator and the child domains regarding the VNF Types they can execute. Table 17 specifies the required parameters for the VNF Type Data Aggregation REST API.

Table 17: Data Aggregation REST API required parameters

Name	Type	Description
Zone	String	The name of the zone.
VNF	String	The name of the VNF Type that can be executed in the zone.
Gateway	String	The Gateway used to connect to the parent zone.
Cost	Float	The monetary cost to execute the VNF in the zone.
Delay	Integer	Network propagation delay from the Gateway to the parent zone.

Figure 32 (D) illustrates the process of updating the Data Aggregation Table. The table is updated if the new data received from the VNF Type Operator provides a better execution option for a given VNF Type compared to the stored data. Additionally, the

table is also updated if the received data introduces a newly available VNF Type.

If the Aggregation Table is updated, a request with the new data to the Data Aggregation API of the parent domain is executed. This process continues until there is no parent domain or the received data does not alter the parent domain's Data Aggregation Table. Figure 32 (E) illustrates this process. With this approach, we ensure that information about the new VNF Type is propagated throughout the domain topology.

6.2.2.1 Data Aggregation Running Example

In this section, we present a running example illustrating the data aggregation process within the Aggregation Zones. Figure 33 (A) depicts a topology with four clusters, each containing an aggregation zone: A1, A2, A3, and A4. Aggregation zone A1 acts as the parent zone for A2 and A3, while A3 serves as the parent zone for A4. These hierarchical relationships allow our proposed approach to analyze the environment and ensure that any changes in VNF availability within the child zones are reported to their respective parent zones. The compute zone was omitted for the sake of simplicity.

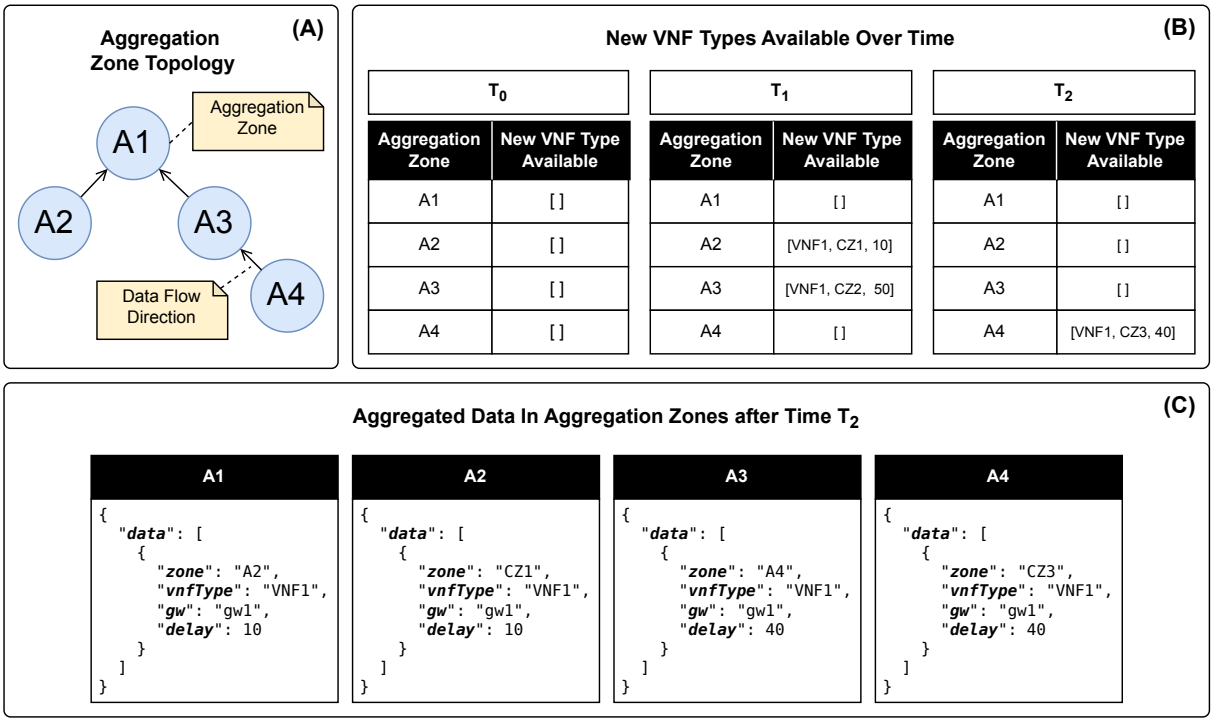


Figure 33: Data Aggregation Process Running Example.

Figure 33 (B) illustrates the availability of VNF Types over time in each aggregation zone. At time T_0, no VNF Types are available in any zone. At time T_1, VNF Type *VNF1* becomes available in aggregation zones A2 and A3. At this point, the Data Aggregation

API of each zone transmits the availability data to its parent zone. Since the data from zone A3 has the same cost but lower latency compared to zone A2, the data from A3 is the one stored in the data aggregation table of A1.

At time T_2, VNF Type *VNF1* becomes available in aggregation zone A4. This information is sent to its parent zone, A3. The new data from A4 has a lower cost than the previously stored data in A3; thus, the aggregation table in A3 is updated accordingly, and the Data Aggregation API of its parent zone (A1) is also triggered. However, in this case, A1 does not update its data aggregation table since the new value from A4 is worse than the previously stored data in A1.

Figure 33 (C) illustrates the aggregated data stored in each aggregation zone after time T_2. The data stored in A1 indicates that the best option for executing a VNF of Type *VNF1* is zone *A2*. Similarly, the data stored in A3 shows that the best option for executing a VNF of Type *VNF1* is zone *A4*. Thus, depending on which zone is selected for conducting the placement of a specific SFC, the selected zones for executing each VNF may vary.

Figure 33 (C) also illustrates the process of hiding information from other domains, a key feature of our proposed approach. The selection of the zone for executing a VNF by the aggregation zone relies solely on the data stored within that zone. Thus, if A1 is queried to determine the appropriate zone for executing a VNF of Type *VNF1*, its response will be A2. However, when A2 is queried, it returns CZ1 as the execution zone for *VNF1*. This means that A1 does not actually know which compute zone will be used for execution; however, it knows the zone that has this information. This strategy enhances the overall security of the environment by minimizing data sharing between domains.

6.2.3 Manager Zone Selection

This section presents the implementation of the manager zone selection from a given SFC request. The manager zone is responsible for coordinating the SFC placement. This process was divided into two algorithms. The first one, presented in Section 4.5.1.2, shows how to search for the SFC request destination zone and, in Section 4.5.1.1, we define the process for selecting the manager zone, taking into account the SFC VNF type requirements and the aggregated data within the zones. By using the output of the first algorithm as input to the second, we can effectively select the SFC request manager zone.

6.2.3.1 Running Example

To facilitate understanding of the manager zone selection process, we provide a running example that illustrates each step in detail. We selected a complex case in which all features of the algorithm are involved. Figure 34 (A) presents the environment where the running example is executed. Figure 34 (B) presents the available VNF Types in each Aggregation zone.

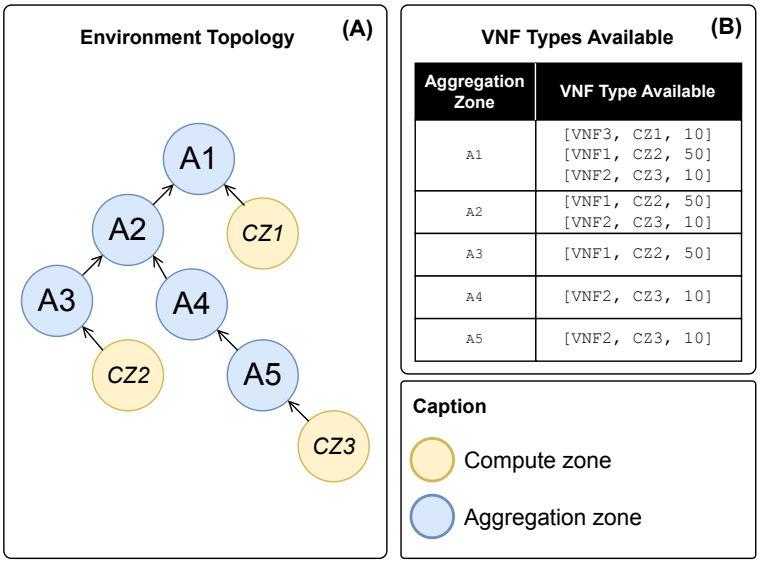


Figure 34: Running example environment.

In this running example, $r_1 = \{CZ3, CZ2, VNFS_1, BW_1, 50, [], t3\}$ is the SFC request. The source zone is $CZ3$, and the destination zone is $CZ2$. The required VNF sequence is $VNFS_1 = [VNF1, VNF2, VNF3]$, and the associated bandwidth requirements are $BW_1 = [vl1 : 10, vl2 : 20]$. The request arrives in the system at time $t3$. There are no constraints regarding compute zones that must be avoided due to political restrictions.

The **first** step of the process is executed by the aggregation zone where the source zone is associated, in this case $A5$. In this step, the aggregation zone checks whether the *dst* zone is among its own compute zones. As the search fails, it then verifies whether there are any child aggregation zones. Since none exist, the request is forwarded to the parent zone $A4$. The **second** step performed in $A4$ occurs in the same way, and the request is forwarded to $A2$.

The **third** step of the process is executed by $A2$, which checks whether the *dst* zone is among its own compute zones. As the search fails, it proceeds to verify the existence of child aggregation zones. In this case, there are two: $A3$ and $A4$. Since $A4$ has already

been explored, the request is forwarded only to $A3$.

The **fourth** step of the process is executed by $A3$, which checks whether the dst zone is among its own compute zones. As the search is successful, it notifies $A2$ that the destination zone is located in $A3$. Finally, $A2$ notifies $A4$, which in turn notifies $A5$ that the destination zone has been found following the path $rzc = [A5, A4, A2, A3]$.

After the destination zone has been found, the process of selecting the manager zone begins in zone $A5$. It verifies that the next position in the rzc variable corresponds to its parent zone, so the request is forwarded to $A4$. Similarly, zone $A4$ checks that the next position in rzc matches its parent zone and forwards the request to $A2$. After verification, zone $A2$ determines that the next position in rzc does not correspond to its parent zone; therefore, it removes all subsequent zones from the rzc list.

Zone $A2$ is now selected as a probable manager zone because both the source and destination zones belong to its sub-tree. It then checks whether all required VNF types are available in its aggregated data. Since the VNF type $VNF3$ is missing, the request is forwarded to $A1$, which confirms that all requested VNFs are available and is therefore selected as the manager zone. The final step is to recursively notify zone $A5$ that zone $A1$ has been selected as the manager zone for the requested SFC.

6.2.4 Distributed SFC Placement

In this section, we present the implementation of our proposed Distributed SFC Placement Approach, named SPEED. Figure 35 illustrates the main activities performed by the components involved in the distributed placement process within the environment. Each domain in the environment runs one instance of each component shown in the figure, enabling parallel and coordinated execution across domains.

The first step of the process consists of receiving the SFC request from the user. The request arrives in the domain where the user is associated. After that, the request will be sent recursively to the parent zone until it reaches a domain that can access the destination zone of the request. This process is executed by the Zone Selection REST API component.

When the request arrives in a domain that can reach the destination zone, it is selected as the coordinated domain. Figure 35 (A) presents the moment when the domain is selected as the coordinated domain. Figure 35 (B) occurs when the root domain can not reach the destination zone, thus the placement fails.

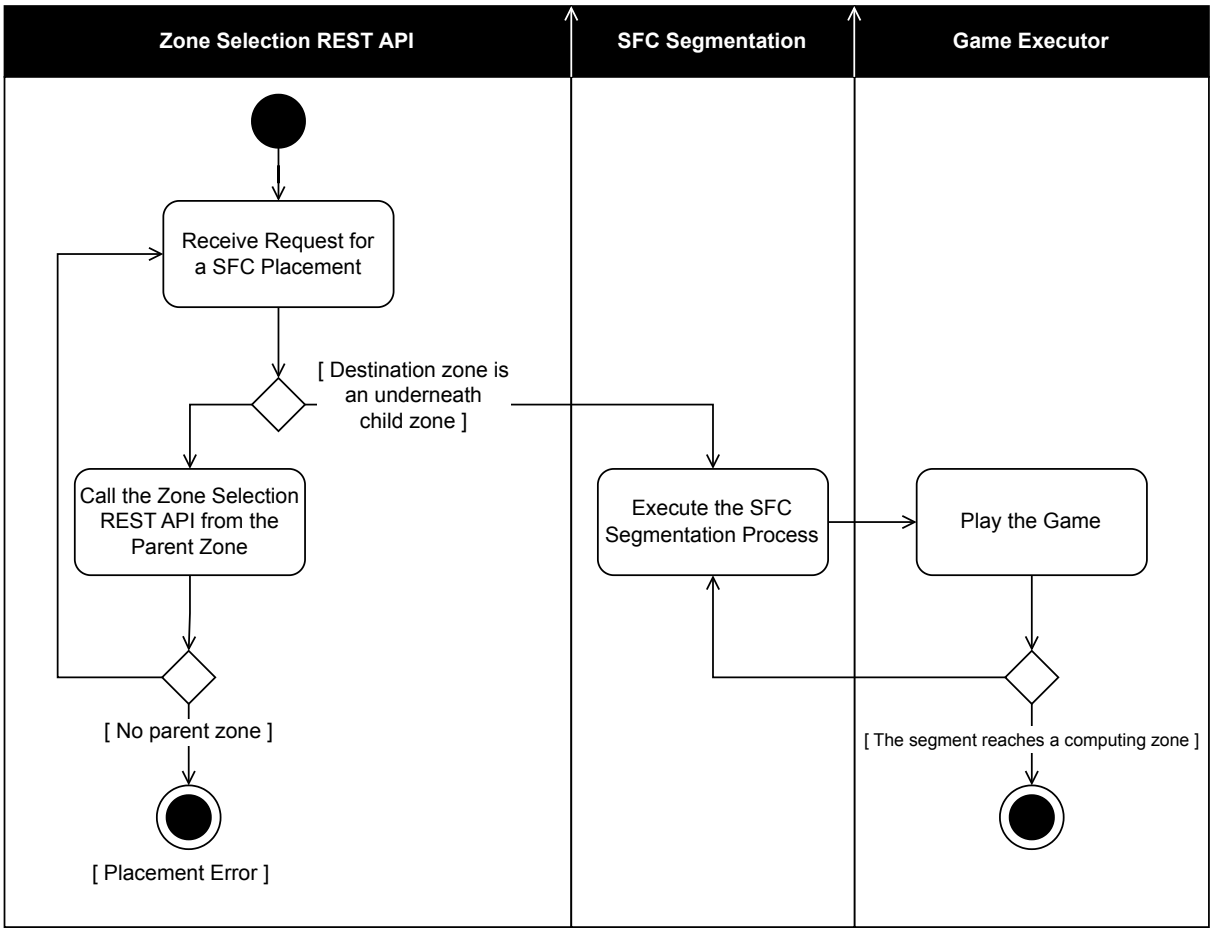


Figure 35: Activity Diagram of the Distributed SFC Placement.

Figure 35 (C) represents the SFC Segmentation process. This process was extensively discussed in Section 4.5.2. In summary, it consists of splitting the VNFs of the SFC into segments, where each segment can be executed in a different domain.

Figure 35 (D) represents the selection of which domain will execute each segment previously created. This process was extensively discussed in Section 4.5.3. Steps (C) and (D) will occur continuously until all the VNFs of SFC are allocated in a domain. In the next sections, we present two representative examples demonstrating the execution of the game under different conditions.

6.2.4.1 Game Execution Running Example (Simple Scenario)

To facilitate a better understanding of how the game is executed, we present a running example that illustrates the detailed steps of the process. This example demonstrates the core features of the algorithm as it generates an SFC Placement Plan from a given SFC request. Figure 36 shows the detailed topology of the environment and the aggregated

data in the aggregation zones.

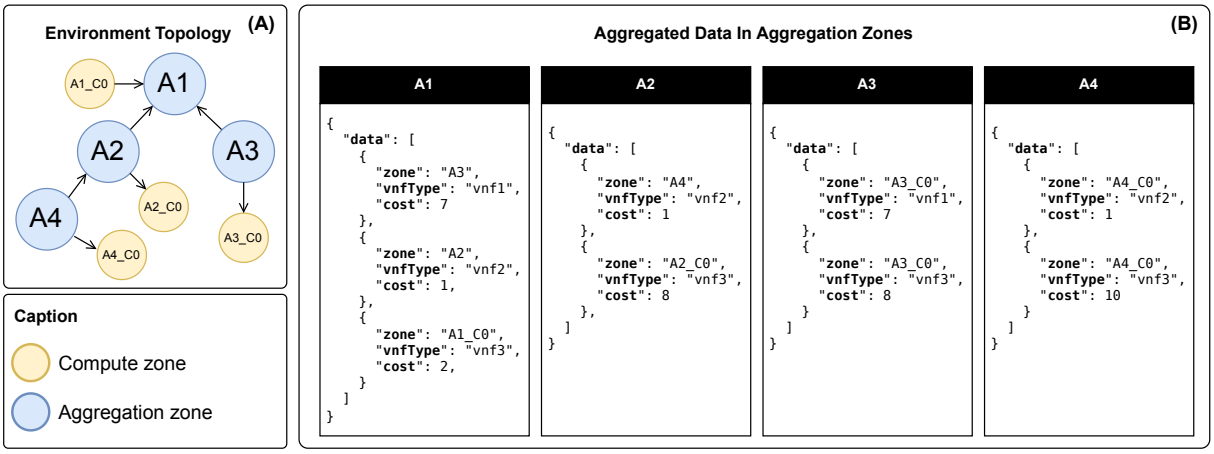


Figure 36: Game Execution Running Example.

Figure 36 (A) presents the topology used in the example execution. The scenario includes four aggregation zones, each representing a domain, and four compute zones, which correspond to the compute nodes responsible for executing the VNFs. Each compute zone is capable of executing a specific set of VNFs, and this information is aggregated and propagated upstream through the topology.

Figure 36 (B) illustrates the aggregated data available at each aggregation zone. The environment supports the execution of three VNF types: `vnf1`, `vnf2`, and `vnf3`. From the perspective of aggregation zone `A1`, the preferred execution zones for each VNF type are as follows: `vnf1` in zone `A3`, `vnf2` in zone `A2`, and `vnf3` in zone `A1_C0`.

The process begins with the arrival of an SFC request at the aggregation zone `A1`, which is the manager zone of this SFC request. The SFC request demands the execution of three VNFs: `vnf1`, `vnf2`, and `vnf3`. Multiple games are executed within `A1` to explore different segmentation strategies until a suitable segmentation plan is identified.

Table 18 presents the segmentation plans generated based on the VNFs requested. The first plan consists of a single segment containing all the VNFs; the second and third plans divide the VNFs into two segments, while the fourth plan splits them into three separate segments. The games corresponding to the first three plans fail because no zone is capable of executing all the VNFs within at least one of their respective segments. In contrast, the game for plan *number four* results in a valid segmentation, as each segment can be executed by at least one available zone.

The game for the segmentation plan *number four* begins where each VNF is separated in its own segment. In the first iteration, the algorithm computes the cost of executing

Table 18: SFC Segmentation Plans in Zone A1

Plan	seg_1_A1	seg_2_A1	seg_3_A1
1	[vnf1, vnf2, vnf3]	–	–
2	[vnf1, vnf2]	[vnf3]	–
3	[vnf1]	[vnf2, vnf3]	–
4	**[vnf1]**	**[vnf2]**	**[vnf3]**

each segment across the available zones. For seg_1_A1, the cheapest execution is found in zone A3; for seg_2_A1, in zone A2; and for seg_3_A1, in zone A1. Penalties are then updated uniformly across all zones. In the second iteration, the costs and penalties remain stable, leading the system to reach equilibrium with no further changes in strategy.

As all segments are successfully assigned to compatible zones, the game in zone A1 terminates and delegates the execution as follows: segment seg_1_A1 to A3, seg_2_A1 to A2, and seg_3_A1 to A1. The game execution in zone A3 is straightforward, as its compute zones are capable of handling all the requested VNFs. In contrast, the execution in zone A2 requires additional games once the zone that can handle the VNF is a child zone of A2.

The game execution in zone A2 begins for the segmentation plan with only one segment containing vnf2. In the first iteration, the algorithm evaluates the cost of executing vnf2 in the local compute zone of A2 and in the aggregation zone A4. The local execution in A2 is not possible, while A4 returns a total cost of 1.0.

After that, the algorithm selects the zone A4 as the cheapest zone to execute the segment. Subsequently updating penalties, the system reaches equilibrium again in the second iteration, with no change in strategy. As the allocation remains stable and all segments are successfully mapped, the algorithm concludes the game and delegates segment seg_0_A2 containing vnf2 to zone A4. Finally, the segment with the vnf2 can be executed in A4 and this zone is selected for executing this segment, returning the information about which zone will execute each VNF to the manager zone.

6.2.4.2 Game Execution Running Example (Complex Scenario With Penalty)

In this section, we present another running example to provide a more comprehensive understanding of how the proposed approach operates in the presence of congestion. This example specifically highlights the role of the penalty mechanism during the game. Figure 37 illustrates the detailed topology of the environment and the aggregated data available at each aggregation zone where the game is executed.

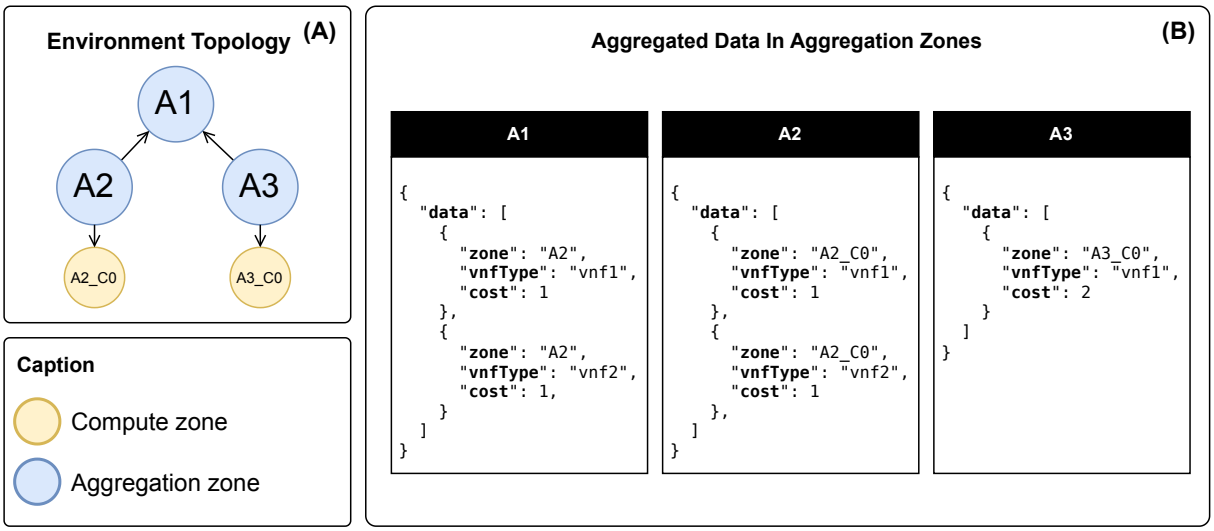


Figure 37: Game Execution Running Example With Penalty.

Figure 37 (A) presents the aggregation zone topology, composed of three aggregation zones (`A1`, `A2`, and `A3`) and two compute zones (`A2_C0` and `A3_C0`). In the topology, compute zone `A2_C0` is connected to aggregation zone `A2`, and `A3_C0` is connected to `A3`. Figure 37 (B) shows the aggregated data available in each aggregation zone regarding the execution cost of different VNFs. Aggregation zone `A1` has in the aggregated data the costs for `vnf1` and `vnf2` in `A2` because the cost of `vnf1` in A3 is higher than `A2`. Zone `A2`, through `A2_C0` can execute both `vnf1` and `vnf2` at a cost of 1. Zone `A3`, via `A3_C0`, can execute only `vnf1`, at a higher cost of 2. This aggregated information is used during the game to determine the allocation of VNFs, considering both execution cost and penalty propagation.

The game starts with a request composed of two VNFs `vnf1` and `vnf2`. These VNFs can be arranged into two segmentation plans. Plan 1 is composed of two segments `[seg_1_A1, seg_2_A1]` where `seg_1_A1=[vnf1]` and `seg_2_A1=[vnf2]`, and plan 2 is composed of only one segment containing both VNFs. The algorithm selects the first plan and begins execution by evaluating each segment independently.

In the first iteration, segment `[seg_1_A1]` cannot be executed locally in zone `A1`. The costs from other aggregation zones are requested, and zone `A2` offers the lowest execution cost. As a result, `seg_1_A1` is initially assigned to `A2`, and the penalty of using `A2` is increased. Then, segment `seg_2_A1` is evaluated. Zone `A2` is again selected, even with an increased penalty, since it remains the only valid zone with the VNF Type requred. Thus, another penalty increment is applied to `A2`.

In the second iteration, the higher penalty accumulated by `A2` affects the decision for

`seg_1_A1`. Although `A2` initially had the lowest cost, the penalty increases its total cost, making `A3` a better option. Therefore, `seg_1_A1` is now assigned to `A3`. The assignment of `seg_2_A1` remains unchanged, as `A2` is still the only available zone for execution. In the third iteration, the assignments remain the same, and the system reaches equilibrium. No further changes occur, and the game in zone `A1` terminates.

Finally, zone `A1` delegates segment `seg_1_A1`, containing `vnf1`, to zone `A3`, and segment `seg_2_A1`, containing `vnf2`, to zone `A2`. Subsequent games are executed in the respective zones, allowing each to select the most appropriate child zone for execution. This example demonstrates that penalty propagation between segments within the same segmentation plan effectively guides the game toward efficient decisions based on the execution cost and congestion of the environment.

6.3 Evaluation

In this section, we present the evaluation of our proposed solution, implemented using components deployed in a real distributed environment. Section 6.3.1 presents the performance evaluation of the aggregation algorithm by analyzing the total amount of data shared and the number of API requests exchanged between domains, in comparison to baseline approaches. Section 6.3.2 depicts the results of the zone selection process. Section 6.3.3 presents the SFC placement success rate obtained by the proposed SPEED approach in comparison with the Auction-based strategy.

6.3.1 Meta Data Aggregation

In this section, we present the results obtained from experiments conducted to compare the proposed **Aggregated approach** against two baseline strategies. The first is a **Non-aggregated approach**, in which the ascendant domains receive information directly, without aggregation, resulting in redundant data dissemination across multiple clusters. The second strategy employs a **Full Mesh approach**, in which every domain receives information from all other domains, creating a fully interconnected network.

To evaluate the proposed solution against the baseline, we conducted a series of experiments. In these experiments, the **independent variables** are i) the *Number of VNF Type Changes*, which indicates how many VNF types become available in the environment, and ii) the *Number of Aggregation Zones*, referring to the number of aggregation zones that compose the environment. The **dependent variables** are i)

the *Number of API Calls*, representing how many times the API responsible for executing the distributed aggregation process is invoked, this metric evaluates the CPU usage to find the placement plan, and ii) the *Data Transferred*, which measures the total amount of data exchanged between the aggregation zones.

The experiments were conducted in heterogeneous environments consisting of multiple independent clusters. To evaluate the proposed approaches, we conducted experiments in environments with different numbers of aggregation zones [5, 8, 10, 13, 15]. Figure 38 illustrates the hierarchical topology of the environment with 15 domains. The other environments follow the same pattern, adopting a complete binary tree topology. These environments closely emulate the real-world Edge-Cloud continuum, providing a realistic context for evaluating the effectiveness of the proposed aggregation approach.

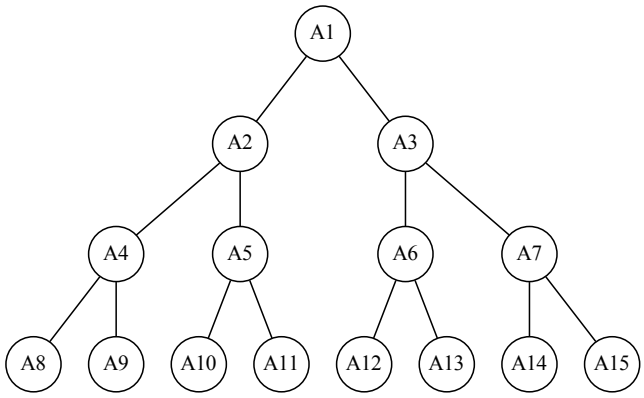


Figure 38: Hierarquical topology domain.

During the experiments, we also varied the numbers of VNF Type changes [50, 100, 200, and 500]. A VNF Type change occurs when a VNF Type becomes available or unavailable in a given cluster. In the experiments, the set of available VNF Types within each domain varies dynamically over time, requiring the propagation of updates to other clusters in the distributed environment. Whenever a new VNF type becomes available within a domain, the SPEED API is invoked to communicate this update to the other clusters. This mechanism ensures timely information sharing among domains.

Figure 39 depicts the experiments comparing our aggregation approach with two baseline approaches: non-aggregated and full-mesh. The results demonstrate that aggregation significantly reduces inter-domain data transfer, especially in larger clusters and scenarios with frequent VNF type changes. This highlights the scalability and efficiency of our approach in real-world, multi-domain environments.

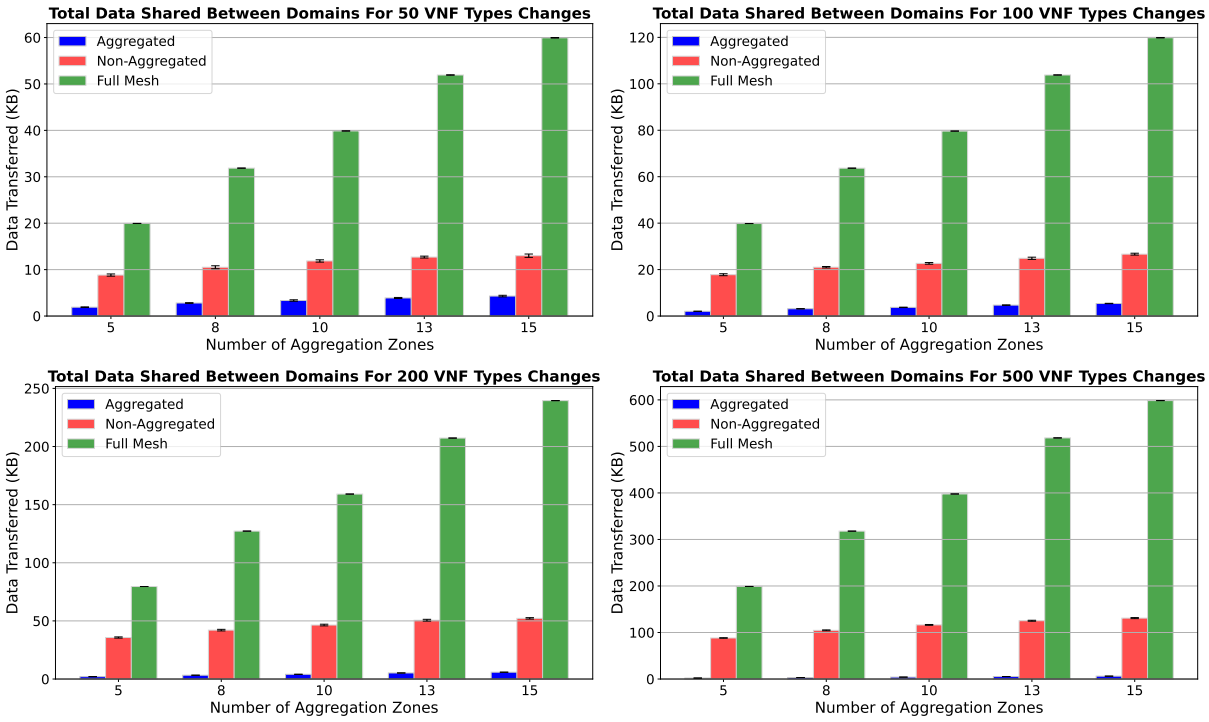


Figure 39: Total Data Shared Between Domains.

Figure 40 depicts a comparison of API calls between domains across different cluster sizes and varying numbers of VNF type changes. Results clearly indicate that the proposed aggregation approach significantly reduces inter-domain API calls compared to the other approaches, especially as both the cluster size and the number of VNF changes increase. The full mesh approach, by comparison, scales poorly, generating substantially higher API calls due to direct communications among all domains, whereas the non-aggregated approach presents intermediate results.

6.3.2 Manager Zone Selection

In this section, we present the results related to the process of identifying the destination zone for SFC requests. Specifically, we evaluate the number of requests made to the SPEED API to determine whether a compute zone is associated with a given aggregation zone. This step is performed during the manager zone selection process, as it consumes computational resources and increases the overall latency of the procedure; therefore, fewer requests lead to a more efficient SFC placement.

This experiment was conducted using the same environment depicted in Section 6.3.1. The **independent variables** consists of i) *Number of SFC Requests*, which indicates how many SFC requests arrived in the environment, and ii) *Number of Aggregation*

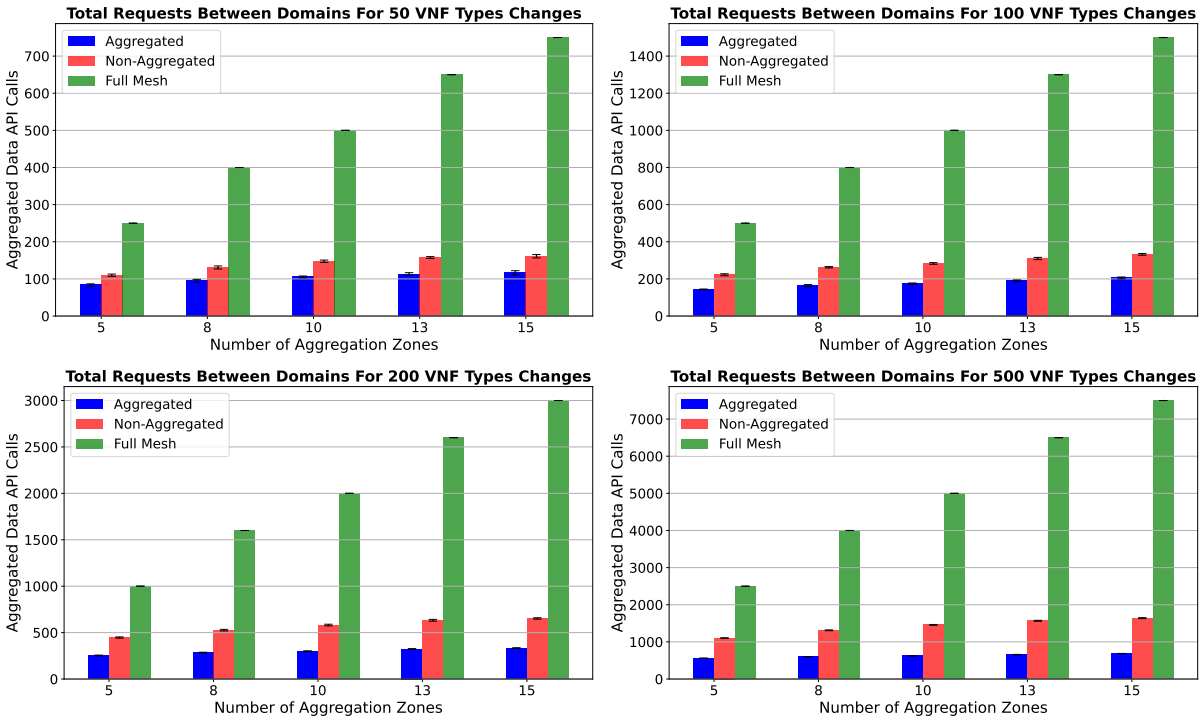


Figure 40: Total Requests API Calls Between Domains.

Zone, referring to the number of aggregation zones that compose the environment. The **dependent variable** comprises the *Number of Search Zone API Calls*, representing how many times the API responsible for executing the search for a compute zone was invoked. By varying the number of aggregation zones and SFC requests, we can analyze how the proposed solution performs under different conditions.

In the experiment, we configured the SPEED API using two different strategies. In the first approach, **Without Early Termination**, the exploration of an aggregation zone is made only if the parent node indicates that the search has not yet been completed. In the second approach, **With Early Termination**, we introduced a distributed mechanism to track whether a search has already been completed by any aggregation zone. This enables more precise control over when an aggregation zone should be explored, thus avoiding unnecessary expansions.

Figure 41 depicts the results of this experiment. Across all tested scenarios, the approach **With Early Termination** consistently required fewer API calls than the **Without Early Termination** configuration. This performance gain becomes significant as the number of SFC requests and the number of aggregation zones increases.

For instance, in the scenario with 500 SFC requests and 15 aggregation zones, the number of API calls **With Early Termination** approach reduced the number of API

calls by approximately 15%. These results confirm that the ability to detect previously completed searches and avoid redundant zone expansions is beneficial for optimizing efficiency in large-scale deployments. This reduction can also contribute to a decrease in SFC placement time as well as a reduction in the usage of compute resources.

It is important to note that in all the scenarios and executions, the selected manager aggregation zones were exactly the same. Besides, regarding the trade-off, this gain comes at the cost of adding a new component responsible for maintaining search data within a defined time frame. This component increases the overall operational overhead and adds a potential point of failure to the system. Therefore, while early termination proves effective in reducing API usage, its adoption must be carefully evaluated in light of fault tolerance and complexity. We conclude that techniques aimed at reducing dependence on the SPEED API are promising, but must balance efficiency gains against architectural considerations.

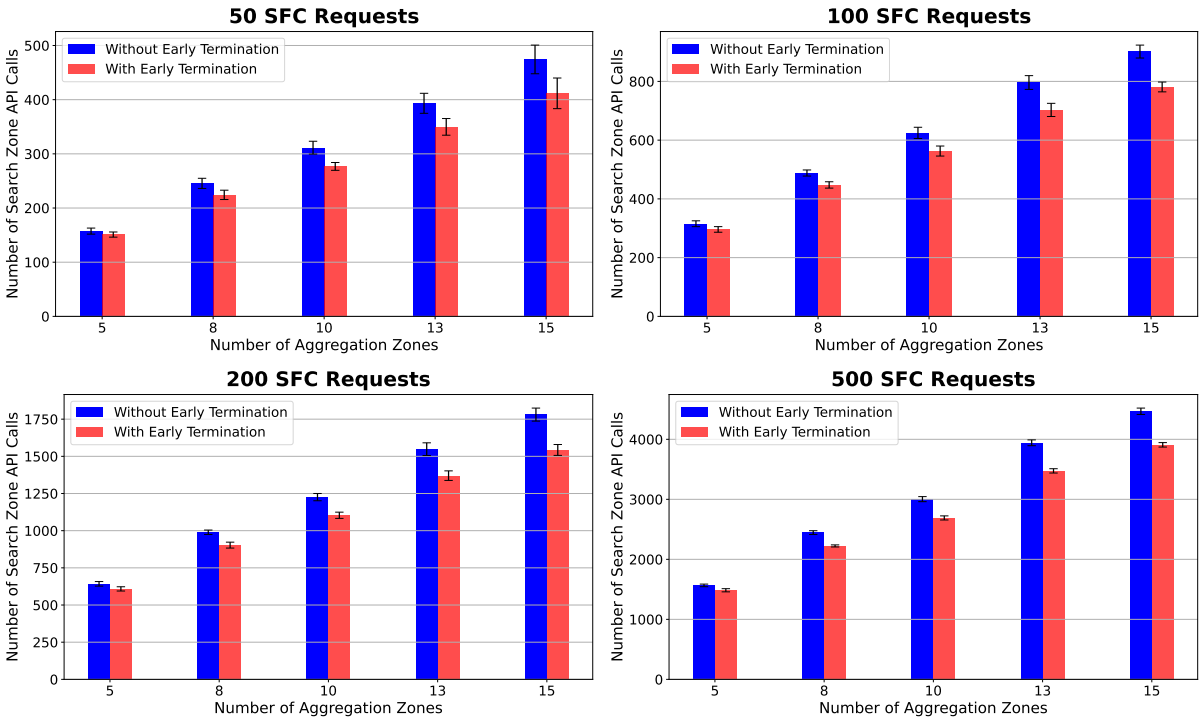


Figure 41: Destination Zone Search Number of SPEED API Calls.

6.3.3 SFC Placement

In this section, we present the results of experiments comparing the proposed **SPEED** approach with a placement strategy related to the auction-based approach described in (AVASALCAI et al., 2019; MACEDO et al., 2023). The auction strategy employs a blind-auction following the first-price sealed-bid format, where each SFC request is mapped to the corresponding service provider domain. In this context, the domain that initiates the auction is the Manager Zone, and the participating domains are those located along the path between the source and the destination zone.

This experiment was conducted using the same environment described in Section 6.3.1. The **independent variable** is the *Number of SFC Requests*, which represents how many requests arrive in the environment, while the **dependent variable** is the *SFC Request Success Rate (%)*, which corresponds to the percentage of requests successfully placed. By varying the number of aggregation zones and the number of SFC requests, we assess how the proposed solution performs under different conditions.

For this experiment, we designed scenarios in which, for every SFC request, a viable solution for the allocation was always available. This means that all SFC requests could be placed given the resources present in the environment. Such a setup allows us to analyze the algorithm's capability to identify and select an available solution.

Figure 42 presents the SFC placement success rate achieved by the proposed SPEED approach compared with the Auction-based strategy under different numbers of domains and varying SFC request volumes. In all scenarios, SPEED consistently outperforms the Auction strategy, maintaining a higher success rate regardless of the number of domains. The performance gap becomes more evident as the number of SFC requests increases, highlighting SPEED's superior capability to identify feasible allocation plans in multi-domain environments.

The results show that, as the number of aggregation zones increases while the compute nodes where each VNF type remains fixed, the probability of selecting a feasible set of zones to execute the SFC decreases. This limitation reduces the flexibility of the placement process and directly lowers the success rate observed in the experiments. The effect becomes stronger with more SFC requests, since fewer combinations of zones remain viable to execute the SFC.

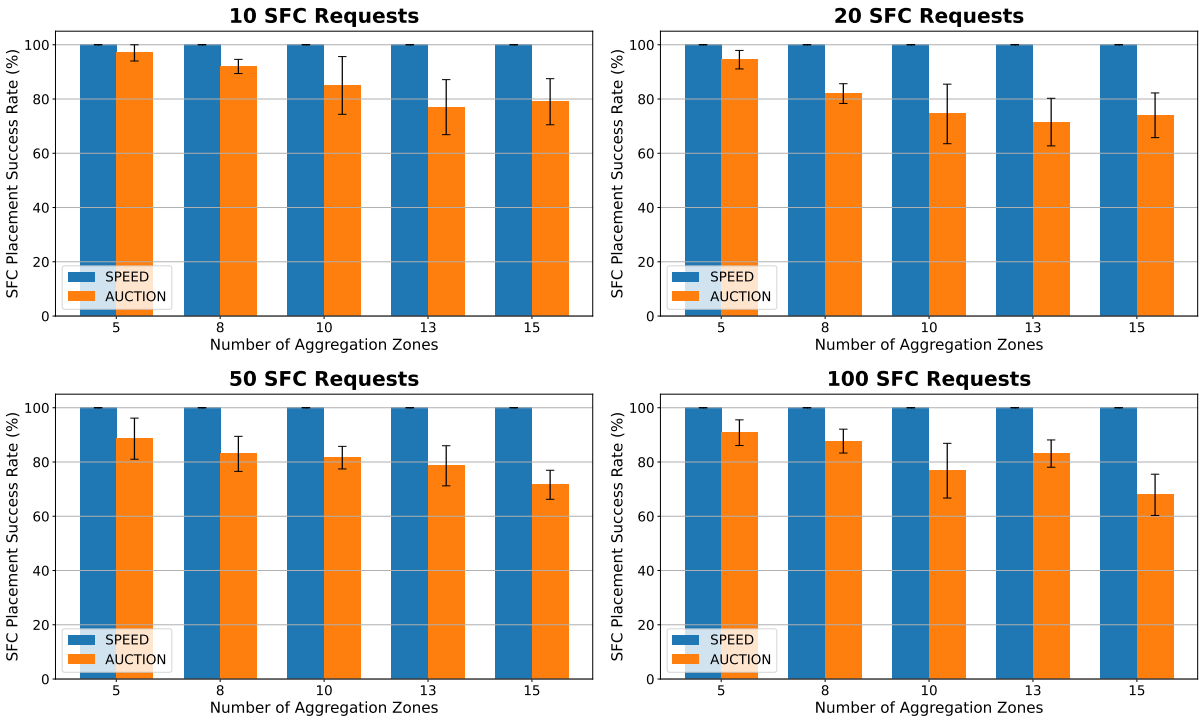


Figure 42: Comparison of the SPEED approach with an Auction-based strategy.

6.4 Lessons Learned

The tools and frameworks used in the process should also be evaluated based on long-term support rather than using a trend tool. Sometimes, a library becomes deprecated or abandoned, which can lead to a deadlock situation where a new tool cannot be integrated into the work. This happens because the dependency on the outdated library prevents the adoption of newer versions of essential tools.

In this work, our proposed approach was based on a multi-domain environment. Running simulations in this scenario was challenging due to the significant hardware requirements. Furthermore, the software used to simulate the multi-domain environment was limited and poorly documented due to the fact that the software is in constant change.

6.5 Conclusion

In this chapter, we evaluated the proposed solution in a real environment. We presented running examples of each step of the process, including data aggregation, manager zone selection, and game execution. Additionally, we provided the results of a series of experiments, covering aspects such as inter-domain data sharing and manager zone selection performance. In the next chapter, we present our final remarks and the

conclusion of this thesis. We will also discuss the limitations of the work, lessons learned, and directions for future research. Finally, we will include a list of publications produced during the development of this PhD.

7 Conclusion

In this thesis, we presented a new solution to the SFC placement problem in multi-domain environments using a distributed approach based on game theory. The proposed solution comprises a set of algorithms that address the key aspects of the problem. We also introduced a new architecture to support the execution of SFCs in such environments.

We conducted a series of experimental evaluations using both simulated and real scenarios. Several running examples were provided to illustrate the behavior of the proposed approach. The results demonstrate that our solution is a viable approach to solving the SFC placement problem in multi-domain environments.

7.1 Contributions

The main contribution of this work is a novel approach, named SFC Placement in Edge-Cloud Continuum (SPEED). The proposed approach solves the SFC placement problem in multi-domain environments using a distributed strategy.

The SPEED approach required the development of a novel architecture that enabled the execution of SFCs across multiple domains. We also presented a new system model that mapped the SFC Placement Problem as a game. This model enabled the resolution of the SFC placement through game theory techniques.

Another contribution of this thesis was a novel strategy for structuring metadata about the VNFs available in the domains. This approach enabled the sharing of VNF metadata information across zones in distributed environments.

Finally, we proposed a new method for solving the SFC Segmentation Problem in distributed environments. This process enabled the VNFs of an SFC to be segmented in a distributed fashion during the SFC placement, allowing parallel execution.

7.2 Answering the Research Questions

In this work, we presented a series of research questions, as follows.

Q1: How can Game Theory be used to solve the Service Function Chain Placement Problem (SFCPP) in a multi-domain Edge-Cloud continuum?

To address this question, we provided a comprehensive description of the proposed solution named SPEED. We formalized the modeling of the SFC Placement Problem as a Singleton Congestion Game. Furthermore, we detailed the algorithms responsible for identifying the manager zone and detailed the processes by which the games are executed. Finally, we presented the architecture necessary to support the execution of our proposed approach.

We presented two different running examples that described the process of creating the SFC Placement in two distinct scenarios. In the first example, we focused on presenting a scenario without congestion. In the second example, we presented the placement of an SFC where congestion occurs.

In the first running example, we were able to show how multiple games and SFC segmentations lead to the selection of which zone will execute each VNF. These games were executed in parallel among multiple domains. We also showed that the creation of the placement plan stops when all VNFs have been assigned to a compute zone.

In the second example, we focused on another key aspect of our proposed approach, which was addressed via the Singleton Congestion Game. This example showed how the system adapts the selection of the zone to execute a segment based on the choices made for other zones. The change in selection, triggered by the selection of the same zone by another segment, demonstrates that the approach works.

Q2: How does the proposed approach contribute to reducing the monetary cost of executing SFCs in a multi-domain Edge-Cloud continuum?

Reducing the monetary cost of SFC execution is a desirable goal for both end users and service providers. From the end user perspective, it implies a lower payment for accessing the service. From the service provider perspective, it may lead to increased profits, as lower prices could stimulate higher demand for SFC deployments.

Reducing the cost of SFC execution is one of the goals we explored. Our proposed approach may contribute to achieving this reduction. However, this objective may involve trade-offs, as lowering monetary costs could negatively affect other performance metrics.

Throughout this thesis, we conducted multiple experiments to evaluate our proposed approach in comparison with existing methods, using the SFC allocation cost as a comparison metric. Simulations demonstrated that our approach was able to find suitable placement plans. However, the execution cost was not the lowest possible when compared to the auction-based method.

Q3: How does the proposed approach contribute to increasing the SFC placement success rate in a multi-domain Edge-Cloud continuum?

Increasing the SFC placement success rate is a central goal of our proposed algorithm, as it directly impacts the system's ability to serve more user requests. SPEED addresses this by leveraging resources across multiple domains, thereby expanding the set of feasible placement options. A higher success rate leads to better resource utilization and improves the overall quality of service.

In this thesis, we presented several experiments to compare our proposed approach with other existing solutions. We ran simulations showing that our approach was 20% less effective than the best solution found via ILP, but 15% more effective than other heuristics in terms of successful placement plan rate. These results demonstrate a good balance between performance and computational efficiency.

7.3 Limitations

Executing a project that proposes multiple algorithms and a comprehensive architecture to solve a complex problem presents several challenges. Throughout this process, we encountered various obstacles that tested our proposed approach. As a result, we were able to identify some limitations and areas for improvement within our solution.

During the execution of the experiments, we utilized only one type of VNF. Although we configured the VNF descriptor to use different amounts of CPU and memory, the code executed by the VNF remained the same. This aspect may have influenced the results observed in some experiments.

Regarding the comparison with other approaches in the literature, we faced significant challenges due to the lack of publicly available source code. When source code was available, it often relied on tools and environments incompatible with our setup. This highlights the need for a common platform to support testing and comparison of SFC placement solutions in distributed environments.

7.4 Future Work

The proposed architecture comprised a set of components required to execute SFCs in a distributed environment. One of these components is a driver that enables communication between our approach and the NFVO-MANO, where the VNFs are executed. In our tests, we only used VNFs in a Network Service Mesh environment. Therefore, additional drivers will be implemented to support SFC execution across multiple domains with distinct NFVO-MANO instances.

During the tests, all domains were Kubernetes clusters running as containers within a virtualized infrastructure. This environment did not fully reflect the network variability typically found in physical Edge-Cloud scenarios. Therefore, we intend to expand the implementation by deploying domains in real cloud infrastructures and on edge devices.

During the execution of this thesis, the cloud-native approach gained significant momentum. This technique refers to building applications for the Edge-Cloud environment with a strong focus on automation and interoperability. We intend to package all components developed in this thesis for cloud-native execution, enabling easy adoption by other researchers.

During the development of our proposed solution based on Singleton Congestion Games, we observed that this approach could also be applied to other tasks related to SFC management. One such task is the migration of a VNF when the provided QoS degrades and becomes inadequate to meet service requirements. In this context, the Singleton Congestion Game can be employed to select a suitable compute zone for executing the VNF that needs to be migrated.

Our approach was mainly focused on a hierarchical solution for sharing data between aggregation zones. We intend to explore a new strategy in which the topology can be more flexible. In such a scenario, the metadata sharing mechanism could be modeled using probabilistic techniques, such as Markov processes, to determine which metadata should be shared with each aggregation zone.

7.5 Research Publications and Developed Projects

This section provides a brief overview of several projects completed during this PhD. In the following sections, we provide a short description of each project and its relevance to this thesis. Figure 43 presents the achievement timeline.

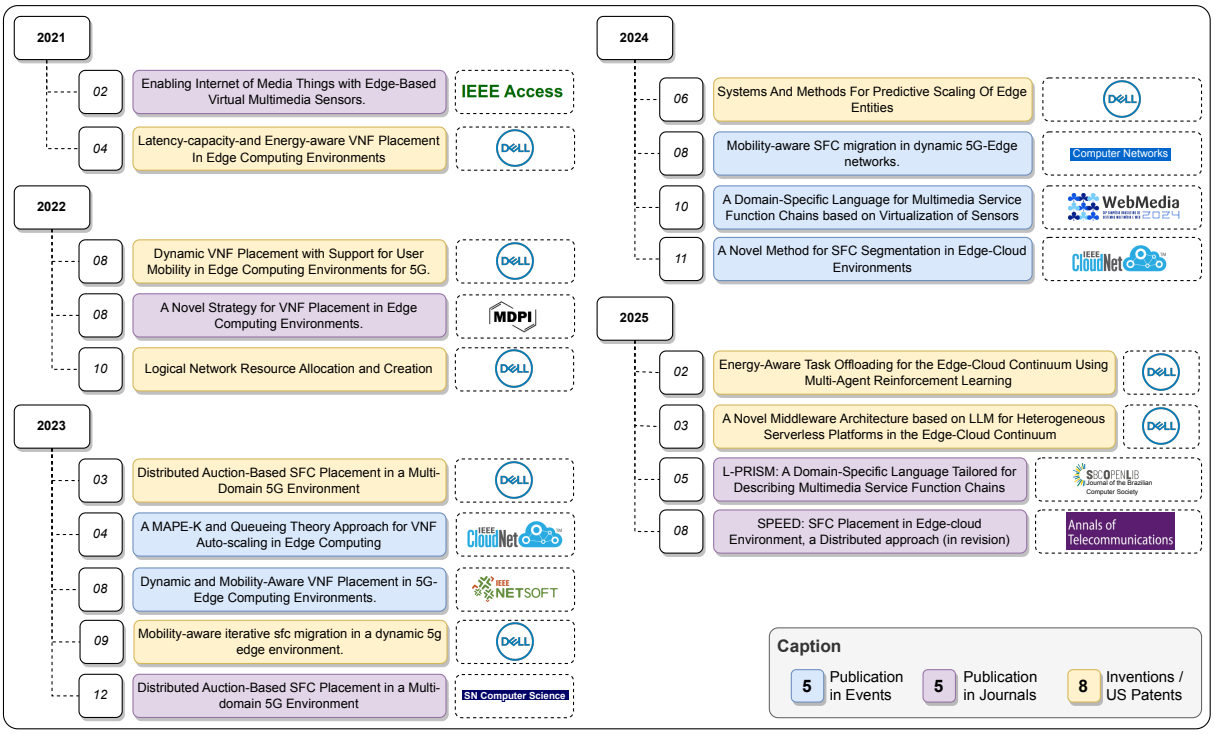


Figure 43: Scientific production timeline.

All of the projects presented provided some insight into this thesis. Section 7.5.1 presents published and submitted articles on VNF placement, distributed VNF and SFC placement, and related topics. Section 7.5.2 shows the inventions developed related to the VNF and VNF placement problem.

7.5.1 Publications

This section presents the publications produced during the development of this thesis. The papers were published in both scientific conferences and journals.

Title: Enabling Internet of Media Things With Edge-Based Virtual Multimedia Sensors
Journal: IEEE Access
Date: 04/21
Status: Published (BATTISTI; MUCHALUAT-SAADE et al., 2021)
Relevance to this thesis:
The idea presented in this paper provided valuable insights into the importance of distributed methods for SFC allocation. It highlighted the inherent limitations of centralized approaches, especially when applied to dynamic and large-scale environments. As a result, the study emphasized the need for efficient and scalable strategies that can adapt to heterogeneous and geographically dispersed infrastructures.

Title: A Novel Strategy for VNF Placement in Edge Computing Environments.
Journal: Future Internet
Date: 08/22
Status: Published | (BATTISTI; MACEDO et al., 2022)
Relevance to this thesis:
This paper served as the foundation for the solution proposed in this thesis. It addresses the VNF placement problem by introducing a novel integer linear programming (ILP) optimization model and a new placement algorithm. In our formulation, the multi-objective optimization problem seeks to simultaneously: i) minimize energy consumption in edge nodes, ii) minimize total latency, and iii) minimize the overall infrastructure cost. The proposed solution extends this formulation by jointly considering these three objectives in the placement process.

Title: Dynamic and Mobility-Aware VNF Placement in 5G-Edge Computing Environments.
Conference: NetSoft 2023
Date: 03/23
Status: Published (VIEIRA; BATTISTI et al., 2023)
Relevance to this thesis:
The ideia presented in this paper helped to understand that the SFC Placement is the begining of the executino of the SFC. After the placement, multiple tasks must be executed to maintain the QoS of the SFC. In this paper, we propose DSP, a Dynamic Smart VNF Placement algorithm that considers the dynamic nature of a 5G edge environment. Our proposal focuses on optimizing resource utilization with an online placement approach to handle continuously arriving service requests. Furthermore, we also account for user mobility by providing an approach that reduces the performance impact caused by user movement.

Title: A MAPE-K and Queueing Theory Approach for VNF Auto-scaling in Edge Computing

Conference: CloudNet 2023

Date: 04/23

Status: Published (SILVA et al., 2023)

Relevance to this thesis:

The ideas presented in this paper helped to understand new concepts, including the MAPE-K approach, which inspired the data aggregation mechanism discussed. In the paper we addressed challenges in scaling VNFs at the edge. We propose a MAPE-K architecture for autonomous auto-scaling using a queue-based VNF scaling algorithm. This algorithm calculates packet processing time probabilities, serving as auto-scaling indicators with scaling thresholds. Scaling up occurs when the probability exceeds a threshold, while scaling down happens below it. The algorithm predicts queue lengths and wait times for VNF instances using a M/M/1 queueing model.

Title: Distributed Auction-Based SFC Placement in a Multi-domain 5G Environment

Journal: SN Computer Science

Date: 12/23

Status: Published (MACEDO et al., 2023)

Relevance to this thesis:

The idea presented in this paper was our initial approach to addressing the SFC placement problem in a distributed manner. We developed a novel solution called the Multi-Domain Distributed Auction-Based SFC Placement Algorithm, which utilizes an auction-based strategy. This approach employs a blind auction, also known as a first-price sealed-bid auction (FPSBA), to associate an SFC request with a service provider domain.

Title: Mobility-aware SFC migration in dynamic 5G-Edge networks.
Journal: Computer Networks
Date: 08/2024
Status: Published (VIEIRA; MACEDO et al., 2024)
Relevance to this thesis:
This paper slightly contributes to the thesis by proposing a proactive and iterative migration strategy for SFCs that mitigates the performance impact of user mobility by anticipating movement and triggering timely migrations. The approach optimizes VNF migration by selecting the VNFs to be migrated and their target nodes, ensuring compliance with service, resource, and migration constraints to maintain quality while minimizing costs.

Title: A Domain-Specific Language for Multimedia Service Function Chains based on Virtualization of Sensors.
Conference: WebMedia 2024
Date: 10/2024
Status: Published (QUICO, Franklin Jordan Ventura et al., 2024)
Relevance to this thesis:
This paper marginally contributes to the thesis by proposing a new language to describe multimedia SFC. This project was an extension of my master degree dissertation. We propose a Domain Specific Language (DSL) called L-PRISM. This DSL can be used as a conceptual base for developers to implement and virtualize multimedia applications using multimedia VNFs.

Title: A Novel Method for SFC Segmentation in Edge-Cloud Environments.
Conference: CloudNet 2024
Date: 11/2024
Status: Published (BATTISTI; DELICATO et al., 2024)
Relevance to this thesis:
This paper makes a significant contribution to the thesis by introducing a novel method for addressing the SFC Segmentation problem in a distributed manner. Our method supports configurable segmentation strategies, allowing various SFC placement methods to customize the segmentation process based on specific requirements. When combined with SPEED, this approach increases the number of SFC requests that can be efficiently placed within a distributed environment.

Title: SPEED: SFC Placement in Edge-Cloud Environment, a Distributed approach.
Journal: Annals of Telecommunications
Date: 08/2025
Status: In progress (Major Review)
Relevance to this thesis:
In this paper, we summarize key contributions proposed throughout this thesis. We introduced the heuristic based on game theory and presented its evaluation in a simulated environment. The results demonstrate the effectiveness of our approach under realistic conditions.

Title: L-PRISM: A Domain-Specific Language for Describing Multimedia Service Function Chains.
Journal: Journal of the Brazilian Computer Society (JBCS)
Date: 05/2025
Status: Published (QUICO, Franklin J. V. et al., 2025)
Relevance to this thesis:
This paper marginally contributes to the thesis by proposing a new language to describe multimedia SFC. This paper presents a new language for describing Multimedia Service Function Chains. It is an extended version of the paper previously published at WebMedia (QUICO, Franklin Jordan Ventura et al., 2024), now including additional use cases and a comprehensive presentation of all language components.

7.5.2 Inventions

This section presents the inventions produced during the development of this thesis. All of these works were carried out in projects funded by DELL in collaboration with multiple researchers. Some of these inventions have been granted as US patents.

Title: Latency-capacity-and Energy-aware VNF Placement In Edge Computing Environments.

Description: In this invention, we addressed the VNF placement problem by providing a novel integer linear programming (ILP) optimization model and a novel heuristic method in order to solve the problem in a non-optimal but faster way. In our definition, the multi-objective optimization problem aims to (i) minimize the energy consumption in the edge nodes, (ii) minimize the total latency, and (iii) reduce the total cost of the infrastructure. In addition, we explore the sharing of VNFs, a feature that is still little explored in the current literature.

Date: 04/21

Project Name: Framework para gerenciamento e execução de Funções de Rede Virtualizadas em infraestruturas 5G.

Company Name: DELL

Patent Application Publication : US20230130420A1

Link: https://patents.google.com/patent/US20230130420A1/en

Title: Dynamic VNF Placement with Support for User Mobility in Edge Computing Environments for 5G.

Description: Address the VNF placement problem by providing a novel graph-based model and a Dynamic Smart VNF Placement algorithm taking into consideration the dynamic aspects of unpredictable requests and user mobility. The new features are i) optimizing resource utilization through an energy-efficient two-level resource sharing mechanism (VNF and SFC Instance sharing); ii) minimizing the cost for the infrastructure provider while respecting QoS requirements; iii) dynamic (online) placement, and iv) user mobility aware placement solution.

Date: 08/22

Project Name: Alocação de recursos distribuída e orientada para a mobilidade em sistemas 5G.

Company Name: DELL

Patent Application Publication : US12143271B2

Link: https://patents.google.com/patent/US12143271B2/en

Title: Logical Network Resource Allocation and Creation.

Description: Addresses techniques for logical network resource allocation and creation. It proposes obtaining a representation of virtualized resources—composed of virtual nodes and virtual links—within a physical network of nodes and links. The method maps virtual nodes to physical nodes and allocates corresponding physical paths for the virtual links, ensuring efficient placement of logical networks over the underlying infrastructure.

Date: 10/23

Project Name: Alocação de recursos distribuída e orientada para a mobilidade em sistemas 5G.

Company Name: DELL

Patent Application Publication : US12244462B2

Link: https://patents.google.com/patent/US12244462B2/en

Title: Distributed Auction-Based SFC Placement in a Multi-Domain 5G Environment.

Description: In this invention, we propose a novel solution for the SFC placement problem called Multi-Domain Distributed Auction-Based SFC Placement Algorithm, which relies on an auction-based strategy. The proposed algorithm targets different 5G application scenarios with focus on solving the SFC placement problem taking into consideration the dynamic aspects of unpredictable requests and the management of the entire auction process to decide which domain will be selected to meet the required SFC.

Date: 03/23

Project Name: Alocação de recursos distribuída e orientada para a mobilidade em sistemas 5G.

Company Name: DELL

Patent Application Publication : US11916742B1

Link: https://patents.google.com/patent/US11916742B1/en

Title: Mobility-aware iterative sfc migration in a dynamic 5g edge environment.
Description: One example method includes predicting a next expected position of the user in a communication network, determining that an SFC request by the user must be migrated from a current node in order to resolve a performance problem of services in the SFC, determining which VNFs of the SFC should be migrated to resolve the performance problem, determining a best migration plan for the VNFs for resolving the identified performance problem, and the best migration plan includes a migration path with a shortest migration delay, finding, in a service chain path identified in the best migration plan, one or more candidate target nodes for migration of the VNFs, and identifying a target node with adequate resources to support the VNFs, and migrating the VNFs from the current node to the target node with the adequate resources.
Date: 09/23
Project Name: Alocação de recursos distribuída e orientada para a mobilidade em sistemas 5G.
Company Name: DELL
Patent Application Publication : US20250106277A1
Link: https://patents.google.com/patent/US20250106277A1/en

Title: Systems And Methods For Predictive Scaling Of Edge Entities
Description: In this invention, we propose A method for scaling a virtual network function (VNF) by analyzing VNF metrics data, the method that includes obtaining, by a computing device, a VNF metrics data, identifying a queue in the VNF metrics data, calculating a constraint violation probability based on the queue, making a first determination that the constraint violation probability is greater than an upper threshold, and based on the first determination, scaling up the VNF. **Date:** 06/24
Project Name: Alocação de recursos distribuída e orientada para a mobilidade em sistemas 5G.
Date: 06/24
Company Name: DELL
Patent Application Publication : US12003378B1
Link: https://patents.google.com/patent/US12003378B1/en

Title: Energy-Aware Task Offloading for the Edge-Cloud Continuum Using Multi-Agent Reinforcement Learning

Description: In this invention, we propose a distributed, multi-agent system that uses the Partially Observable Markov Decision Process (PO-MDP) and Multi-Agent Deep Deterministic Policy Gradient (MA-DDPG) to optimize task offloading decisions based on local information. This decentralized approach allows the system to adapt dynamically, autonomously balancing energy consumption and latency, while ensuring compliance with Service Level Agreements (SLAs). The system is designed to perform effectively even under uncertain network conditions, mobility, and partial observability, where traditional centralized models fail.

Project Name: Plataforma baseada em Serverless e Aprendizado de Máquina para ambientes de Edge-Cloud.

Date: 02/25

Company Name: DELL

Title: A Novel Middleware Architecture based on LLM for Heterogeneous Serverless Platforms in the Edge-Cloud Continuum

Description: This invention introduces a middleware that leverages multiple serverless platforms distributed across Edge-Cloud nodes to execute users requests seamlessly. The middleware is designed to process user requests issued in natural language, and it does so by using Large Language Models (LLMs) to understand the intent behind each request. Once the request is understood, the middleware handles the complexity of mapping it to a specific serverless function, dynamically requesting the execution of the workload in the selected serverless platform whilst keeping transparency for end users.

Project Name: Plataforma baseada em Serverless e Aprendizado de Máquina para ambientes de Edge-Cloud.

Date: 03/25

Company Name: DELL

REFERENCES

ABUJODA, Ahmed; PAPADIMITRIOU, Panagiotis. DistNSE: Distributed network service embedding across multiple providers. In: COMSNETS. Bangalore, India: IEEE, 2016. P. 1–8. DOI: 10.1109/COMSNETS.2016.7439948.

ALAHMAD, Yanal; AGARWAL, Anjali; DARADKEH, Tariq. Cost and Availability-Aware VNF Selection and Placement for Network Services in NFV. In: 2020 International Symposium on Networks, Computers and Communications (ISNCC). Montreal, Canada: IEEE, 2020. P. 1–6. DOI: 10.1109/ISNCC49221.2020.9297190.

ALAM, Iqbal; SHARIF, Kashif; LI, Fan; LATIF, Zohaib; KARIM, M. M.; BISWAS, Sujit; NOUR, Boubakr; WANG, Yu. A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 53, n. 2, abr. 2020. ISSN 0360-0300. DOI: 10.1145/3379444.

AVASALCAI, Cosmin; TSIGKANOS, Christos; DUSTDAR, Schahram. Decentralized Resource Auctioning for Latency-Sensitive Edge Computing. In: 2019 IEEE International Conference on Edge Computing (EDGE). Milan, Italy: IEEE, 2019. P. 72–76. DOI: 10.1109/EDGE.2019.00027.

BATTISTI, Anselmo Luiz Éden; DELICATO, Flavia C.; MUCHALUAT-SAADE, Débora Christina. A Novel Method for SFC Segmentation in Edge-Cloud Environments. en. In: CLOUDNET. 2024 IEEE 13th International Conference on Cloud Networking (CloudNet). Rio de Janeiro, Brasil: IEEE, nov. 2024. P. 1–8. DOI: 10.1109/cloudnet62863.2024.10815829.

BATTISTI, Anselmo Luiz Éden; MACEDO, Evandro Luiz Cardoso; JOSUÉ, Marina Ivanov Pereira; BARBALHO, Hugo; DELICATO, Flávia C.; MUCHALUAT-SAADE, Débora Christina; PIRES, Paulo F.; MATTOS, Douglas Paulo de; OLIVEIRA, Ana Cristina Bernardo de. A Novel Strategy for VNF Placement in Edge Computing Environments. **Future Internet**, v. 14, n. 12, 2022. ISSN 1999-5903. DOI: doi.org/10.3390/fi14120361.

BATTISTI, Anselmo Luiz Éden; MUCHALUAT-SAADE, Débora Christina; DELICATO, Flávia C. Enabling Internet of Media Things With Edge-Based Virtual Multimedia Sensors. **IEEE Access**, v. 9, p. 59255–59269, 2021. DOI: 10.1109/ACCESS.2021.3073240.

BHAMARE, Deval; JAIN, Raj; SAMAKA, Mohammed; ERBAD, Aiman. A survey on service function chaining. **Journal of Network and Computer Applications**, v. 75, p. 138–155, 2016. ISSN 1084-8045. DOI: doi.org/10.1016/j.jnca.2016.09.001.

BILÒ, Vincenzo; VINCI, Carlo. On the Impact of Singleton Strategies in Congestion Games. In: 25TH Annual European Symposium on Algorithms (ESA 2017). [S. l.]: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. v. 87. (Leibniz International Proceedings in Informatics (LIPIcs)), 17:1–17:14. DOI: 10.4230/LIPIcs.ESA.2017.17.

BITTAR, Abdullah; WANG, Ziqiang; AGHASHARIF, Amir; HUANG, Changcheng; SHAMI, Gauravdeep; LYONNAIS, Marc; WILSON, Rodney. Service function chaining design & implementation using network service mesh in kubernetes. In: SPRINGER INTERNATIONAL PUBLISHING CHAM. ASIAN Conference on Supercomputing Frontiers. Singapore: Springer International Publishing, 2022. P. 121–140. DOI: doi.org/10.1007/978-3-031-10419-0_8.

BRAILSFORD, Sally C; POTTS, Chris N; SMITH, Barbara M. Constraint satisfaction problems: Algorithms and applications. **European journal of operational research**, Elsevier, v. 119, n. 3, p. 557–581, 1999. DOI: doi.org/10.1016/S0377-2217(98)00364-6.

BREITGAND, David; LEKIDIS, Alexios; BEHRAVESH, Rasoul; WEIT, Avi; GIARDINA, Pietro; THEODOROU, Vasileios; COSTA, Cristina E; BARABASH, Katherine. Dynamic slice scaling mechanisms for 5G multi-domain environments. In: 2021 IEEE 7th International Conference on Network Softwarization (NetSoft). Tokyo, Japan: IEEE, 2021. P. 56–62. DOI: 10.1109/NetSoft51509.2021.9492716.

BUNYAKITANON, Monchai; VASILAKOS, Xenofon; NEJABATI, Reza; SIMEONIDOU, Dimitra. End-to-End Performance-Based Autonomous VNF Placement With Adopted Reinforcement Learning. **IEEE Transactions on Cognitive Communications and Networking**, v. 6, n. 2, p. 534–547, 2020. DOI: 10.1109/TCCN.2020.2988486.

CHEN, Chen; NAGEL, Lars; CUI, Lin; TSO, Fung Po. Distributed federated service chaining for heterogeneous network environments. **ACM International Conference Proceeding Series**, Association for Computing Machinery, 2021. DOI: 10.1145/3468737.3494091.

_____. Distributed federated service chaining: A scalable and cost-aware approach for multi-domain networks. **Computer Networks**, Elsevier BV, v. 212, p. 109044, jul. 2022. DOI: 10.1016/j.comnet.2022.109044.

CHEN, Hung-Li; LIN, Fuchun Joseph. Scalable IoT/M2M Platforms Based on Kubernetes-Enabled NFV MANO Architecture. In: 2019 International Conference on Internet of Things (iThings). Atlanta, GA, USA: IEEE, 2019. P. 1106–1111. DOI: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00188.

CISNEROS, Josue Castaneda; YANGUI, Sami; HERNANDEZ, Saul E. Pomares; DRIRA, Khalil. A Survey on Distributed NFV Multi-Domain Orchestration from an Algorithmic Functional Perspective. en. **IEEE Communications Magazine**, 2022. ISSN 0163-6804, 1558-1896. DOI: 10.1109/MCOM.002.2100950.

COÊLHO, Roger William; SILVA, Ronan Assumpção; MARTIMIANO, Luciana Andréia Fondazzi; LEONARDO, Elvio João. IoT and 5G Networks: A Discussion of SDN, NFV and Information Security. **Journal of the Brazilian Computer Society**, v. 30, n. 1, p. 212–227, ago. 2024. DOI: 10.5753/jbcs.2024.3021.

DAB, Boutheina; FAJJARI, Ilhem; ROHON, Mathieu; AUBOIN, Cyril; DIQUÉLOU, Arnaud. Cloud-native Service Function Chaining for 5G based on Network Service Mesh. In: ICC 2020 - 2020 IEEE International Conference on Communications (ICC). Virtual Conference: IEEE, 2020. P. 1–7. DOI: 10.1109/ICC40277.2020.9149045.

DOCUMENTATION, Kubernetes. **Extend Kubernetes with Custom Resources**. [S. l.: s. n.], 2024. Accessed: 2024-01-19. Disponível em: <https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/>.

EMU, Mahzabeen; YAN, Peizhi; CHOUDHURY, Salimur. 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). PROCEEDINGS of the 2020 IEEE International Conference on Communications

Workshops (ICC Workshops). [S. l.]: IEEE, 2020. P. 1–6. DOI: 10.1109/ICCWorkshops49005.2020.9145145.

ETSI. **GS MEC 003 - V3.1.1 - Multi-access Edge Computing (MEC); Framework and Reference Architecture**. v. 1. [S. l.], 2022.

_____. **NFV Release 5 Description**. v. 1. Valbonne/França, 2019.

FOUKAS, Xenofon; PATOUNAS, Georgios; ELMOKASHFI, Ahmed. Network slicing in 5G: Survey and challenges. **IEEE Communications Magazine**, IEEE, v. 55, n. 5, p. 94–100, 2017. DOI: 10.1109/MCOM.2017.1600951.

FUDENBERG, Drew; LEVINE, David K. **The Theory of Learning in Games**. Cambridge, MA: MIT press, 1998. v. 2.

GAIRING, Martin; SCHOPPMANN, Florian. Total Latency in Singleton Congestion Games. In_____. **Proceedings of the 3rd International Workshop on Internet and Network Economics (WINE 2007)**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. P. 381–387. ISBN 978-3-540-77105-0. DOI: 10.1007/978-3-540-77105-0_42.

GAO, Tao; LI, Xin; WU, Yu; ZOU, Weixia; HUANG, Shanguo; TORNATORE, Massimo; MUKHERJEE, Biswanath. Cost-Efficient VNF Placement and Scheduling in Public Cloud Networks. **IEEE Transactions on Communications**, v. 68, n. 8, p. 4946–4959, 2020. DOI: 10.1109/TCOMM.2020.2992504.

GAO, Xiangqiang; LIU, Rongke; KAUSHIK, Aryan. Virtual Network Function Placement in Satellite Edge Computing With a Potential Game Approach. **IEEE Transactions on Network and Service Management**, IEEE Press, v. 19, n. 2, p. 1243–1259, jun. 2022. ISSN 1932-4537. DOI: 10.1109/TNSM.2022.3141165.

GARRICH, Miquel; ROMERO-GÁZQUEZ, et.al. IT and Multi-layer Online Resource Allocation and Offline Planning in Metropolitan Networks. **Journal of Lightwave Technology**, v. 38, n. 12, p. 3190–3199, 2020. DOI: 10.1109/JLT.2020.2990066.

HALABIAN, Hassan. Distributed Resource Allocation Optimization in 5G Virtualized Networks. **IEEE Journal on Selected Areas in Communications**, v. 37, n. 3, p. 627–642, 2019. DOI: 10.1109/JSAC.2019.2894305.

HALPERN, Joel; PIGNATARO, Carlos. **Service Function Chaining (SFC) Architecture**. Wilmington, DE / USA: Internet Engineering Task Force, out. 2015. RFC 7665. Category: Informational. DOI: 10.17487/RFC7665.

HUFF, Alexandre; VENÂNCIO, Giovanni; GARCIA, Vinícius Fulber; DUARTE, Elias P. Building Multi-domain Service Function Chains Based on Multiple NFV Orchestrators. In: 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). Virtual Conference: IEEE, 2020. P. 19–24. DOI: 10.1109/NFV-SDN50289.2020.9289849.

HUFF, Alexandre; VENÂNCIO, Giovanni; JR., Elias Duarte. 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. Gramado: SBC, 2019. P. 777–790. DOI: 10.5753/sbrc.2019.7402.

IWASE, Tatsuya; TADOKORO, Yukihiro; FUKUDA, Daisuke. Self-Fulfilling Signal of an Endogenous State in Network Congestion Games. **Networks and Spatial Economics**, Springer Science e Business Media LLC, v. 17, n. 3, p. 889–909, jun. 2017. ISSN 1572-9427. DOI: 10.1007/s11067-017-9351-4.

KAPASSA, Evgenia; TOULOUPOU, Marios; KYRIAZIS, Dimosthenis. 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). Krakow, Poland: IEEE, 2018. v. 1, p. 469–474. DOI: 10.1109/WAINA.2018.00130.

KATSALIS, Kostas; NIKAEIN, Navid; EDMONDS, Andy. Multi-Domain Orchestration for NFV: Challenges and Research Directions. In: 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS). Granada, Spain: IEEE, 2016. DOI: 10.1109/IUCC-CSS.2016.034.

KAUR, Karamjeet; MANGAT, Veenu; KUMAR, Krishan. A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture. en. **Computer Science Review**, v. 38, 2020. ISSN 15740137. DOI: doi.org/10.1016/j.cosrev.2020.100298.

KIM, Sang il; SUNG KIM, Hwa. A VNF Placement Method Considering Load and Hop count in NFV Environment. In: ICOIN. 2020 International Conference on Information Networking (ICOIN). Barcelona, Spain: IEEE, 2020. P. 707–712. DOI: 10.1109/ICOIN48656.2020.9016492.

KIRAN, Nahida; LIU, Xuanlin; WANG, Sihua; YIN, Changchuan. VNF Placement and Resource Allocation in SDN/NFV-Enabled MEC Networks. In: WCNCW. 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). Seoul, Korea (South): IEEE, 2020. P. 1–6. DOI: 10.1109/WCNCW48565.2020.9124910.

KNIGHT, Simon; NGUYEN, Hung X.; FALKNER, Nickolas; BOWDEN, Rhys; ROUGHAN, Matthew. The Internet Topology Zoo. **IEEE Journal on Selected Areas in Communications**, v. 29, n. 9, p. 1765–1775, 2011. DOI: 10.1109/JSAC.2011.111002.

KUO, Tung-Wei; LIOU, Bang-Heng; LIN, Kate Ching-Ju; TSAI, Ming-Jer. Deploying Chains of Virtual Network Functions: On the Relation Between Link and Server Usage. **IEEE/ACM Transactions on Networking**, v. 26, n. 4, p. 1562–1576, 2018. DOI: 10.1109/TNET.2018.2842798.

LAGHRISSI, Abdelquoddouss; TALEB, Tarik. A Survey on the Placement of Virtual Resources and Virtual Network Functions. **IEEE Communications Surveys and Tutorials**, IEEE, v. 21, n. 2, p. 1409–1434, 2019. ISSN 1553877X. DOI: 10.1109/COMST.2018.2884835.

LEE, Whay C. Topology aggregation for hierarchical routing in ATM networks. **Computer Communication Review**, v. 25, n. 2, p. 82–92, 1995. ISSN 01464833. DOI: 10.1145/210613.210625.

LI, Congzhou; WU, Zhouxiang; KHANURE, Divya; JUE, Jason P. A Multi-Agent Reinforcement Learning Scheme for SFC Placement in Edge Computing Networks. In: 2025 International Conference on Computing, Networking and Communications (ICNC). Honolulu, Hawaii, USA: IEEE, 2025. P. 446–451. DOI: 10.1109/ICNC64010.2025.10994017.

LI, Defang; HONG, Peilin; XUE, Kaiping; PEI, Jianing. Virtual network function placement and resource optimization in NFV and edge computing enabled networks. **Computer Networks**, v. 152, p. 12–24, 2019. ISSN 13891286. DOI: doi.org/10.1016/j.comnet.2019.01.036.

LI, Junling; SHI, Weisen; YE, Qiang; ZHANG, Ning; ZHUANG, Weihua; SHEN, Xuemin. Multiservice Function Chain Embedding With Delay Guarantee: A Game-Theoretical Approach. **IEEE Internet of Things Journal**, v. 8, n. 14, p. 11219–11232, 2021. DOI: 10.1109/JIOT.2021.3051905.

LI, Shuopeng; ZHANG, Shaohui; CHEN, Limin; CHEN, Huamin; LIU, Xiliang; LIN, Shaofu. An Attention Based Deep Reinforcement Learning Method for Virtual Network Function Placement. In: 2020 IEEE 6th International Conference on Computer and Communications (ICCC). Chengdu, China: IEEE, 2020. P. 1005–1009. DOI: 10.1109/ICCC51575.2020.9345041.

LI, Taixin; ZHOU, Huachun; LUO, Hongbin. A new method for providing network services: Service function chain. **Optical Switching and Networking**, v. 26, 2017. ISSN 15734277. DOI: doi.org/10.1016/j.osn.2015.09.005.

LI, Xin; QIAN, Chen. A survey of network function placement. In: 2016 13th IEEE Annual Consumer Communications& Networking Conference (CCNC). Las Vegas / USA: IEEE, 2016. P. 948–953. DOI: 10.1109/CCNC.2016.7444915.

LI, Yaru; LI, Lin; BAI, Jing; CHANG, Xiaolin; YAO, Yingying; LIU, Peide. Availability and Reliability of Service Function Chain: A Quantitative Evaluation View. **International Journal of Computational Intelligence Systems**, Springer Science e Business Media LLC, v. 16, n. 1, abr. 2023. ISSN 1875-6883. DOI: 10.1007/s44196-023-00215-8.

LIU, Yi; ZHANG, Hongqi; CHANG, Dexian; HU, Hao. GDM: A General Distributed Method for Cross-Domain Service Function Chain Embedding. **IEEE Transactions on Network and Service Management**, v. 17, n. 3, p. 1446–1459, 2020. DOI: 10.1109/TNSM.2020.2993364.

MACEDO, Evandro L. C.; BATTISTI, Anselmo L. E.; VIEIRA, Juan Lucas; NOCE, Julia; PIRES, Paulo F.; MUCHALUAT-SAADE, Débora C.; OLIVEIRA, Ana C. B.; DELICATO, Flavia C. Distributed Auction-Based SFC Placement in a Multi-domain 5G Environment. **SN Computer Science**, Springer Science e Business Media LLC, v. 5, n. 1, p. 48, dez. 2023. ISSN 2661-8907. DOI: 10.1007/s42979-023-02291-1.

MAKKI, SAM; HAVAS, George. Distributed algorithms for depth-first search. **Information Processing Letters**, v. 60, n. 1, p. 7–12, 1996. DOI: doi.org/10.1016/S0020-0190(96)00141-X.

MAMUSHIANE, Lusani; LYSKO, Albert A.; MUKUTE, Tariro; MWANGAMA, Joyce; TOIT, Zaaid Du. Overview of 9 Open-Source Resource Orchestrating ETSI MANO Compliant Implementations: A Brief Survey. In: 2019 IEEE 2nd Wireless Africa Conference (WAC). Pretoria, South Africa: IEEE, 2019. P. 1–7. DOI: 10.1109/AFRICA.2019.8843421.

MAO, Yuyi; YOU, Changsheng; ZHANG, Jun; HUANG, Kaibin; LETAIEF, Khaled B. A Survey on Mobile Edge Computing: The Communication Perspective. **IEEE Communications Surveys & Tutorials**, v. 19, n. 4, p. 2322–2358, 2017. DOI: 10.1109/COMST.2017.2745201.

MASKIN, Eric. Nash Equilibrium and Welfare Optimality. **Review of Economic Studies**, v. 66, p. 23–38, 1999. Reprinted in J.J. Laffont (ed.), The Principal Agent Model: The Economic Theory of Incentives, London: Edward Elgar, 2003.

MECHTRI, Marouen; BENYAHIA, Imen Grida; ZEGHLACHE, Djamal. Agile service manager for 5G. In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. Istanbul, Turkey: IEEE, 2016. P. 1285–1290. DOI: 10.1109/NOMS.2016.7503004.

MIJUMBI, Rashid; SERRAT, Joan; GORRICHO, Juan-Luis; BOUTEN, Niels; DE TURCK, Filip; BOUTABA, Raouf. Network Function Virtualization: State-of-the-Art and Research Challenges. **IEEE Communications Surveys & Tutorials**, v. 18, n. 1, p. 236–262, 2016. DOI: 10.1109/COMST.2015.2477041.

MOHAMAD, Amir; HASSANEIN, Hossam S. On Demonstrating the Gain of SFC Placement with VNF Sharing at the Edge. In: GLOBECOM. 2019 IEEE Global Communications Conference. Waikoloa, HI, USA: IEEE, 2019. P. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9014106.

_____. PSVShare: A Priority-based SFC placement with VNF Sharing. In: 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). Leganes, Spain: IEEE, 2020. P. 25–30. DOI: 10.1109/NFV-SDN50289.2020.9289837.

MORABITO, Roberto; COZZOLINO, Vittorio; DING, Aaron Yi; BEIJAR, Nicklas; OTT, Jorg. Consolidate IoT Edge Computing with Lightweight Virtualization. **IEEE Network**, v. 32, n. 1, p. 102–111, 2018. DOI: 10.1109/MNET.2018.1700175.

MOSTAFAVI, Seyedakbar; HAKAMI, Vesal. Quality of service provisioning in network function virtualization: a survey. en. **Computing**, v. 103, n. 5, p. 917–991, 2021. ISSN 0010-485X, 1436-5057. DOI: doi.org/10.1007/s00607-021-00925-x.

MOUALLA, Ghada; TURLETTI, Thierry; SAUCEZ, Damien. An Availability-Aware SFC Placement Algorithm for Fat-Tree Data Centers. In: CLOUDNET. Tokyo, Japan: IEEE, 2018. P. 1–4. DOI: 10.1109/CloudNet.2018.8549338.

NGUYEN, Duong Tuan; PHAM, Chuan; NGUYEN, Kim Khoa; CHERIET, Mohamed. Placement and Chaining for Run-Time IoT Service Deployment in Edge-Cloud. **IEEE Transactions on Network and Service Management**, v. 17, n. 1, p. 459–472, 2020. DOI: 10.1109/TNSM.2019.2948137.

NIU, Meng; CHENG, Bo; CHEN, Jun-Liang. GPSO: A Graph-based Heuristic Algorithm for Service Function Chain Placement in Data Center Networks. In: SCC. 2020 IEEE International Conference on Services Computing (SCC). Virtual, Beijing, China: IEEE, 2020. P. 256–263. DOI: 10.1109/SCC49832.2020.00041.

OJO, Mike; ADAMI, Davide; GIORDANO, Stefano. A SDN-IoT Architecture with NFV Implementation. In: 2016 IEEE Globecom Workshops (GC Wkshps). Washington, USA: IEEE, 2016. DOI: 10.1109/GLOCOMW.2016.7848825.

OWEN, G. **Game Theory**. Bingley, UK: Emerald Group Publishing Limited, 2013. ISBN 9781781905074.

PAN, Jianli; MCELHANNON, James. Future Edge Cloud and Edge Computing for Internet of Things Applications. **IEEE Internet of Things Journal**, v. 5, n. 1, p. 439–449, 2018. DOI: 10.1109/JIOT.2017.2767608.

PATTARANANTAKUL, Montida; VORAKULPIPAT, Chalee; TAKAHASHI, Takeshi. Service Function Chaining security survey: Addressing security challenges and threats. **Computer Networks**, v. 221, 2023. ISSN 13891286. DOI: doi.org/10.1016/j.comnet.2022.109484.

PEI, Jianing; HONG, Peilin; PAN, Miao; LIU, Jiangqing; ZHOU, Jingsong. Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks. **IEEE Journal on Selected Areas in Communications**, IEEE, v. 38, n. 2, p. 263–278, 2020. ISSN 15580008. DOI: 10.1109/JSAC.2019.2959181.

PENTELAS, Angelos; DE VLEESCHAUWER, Danny; CHANG, Chia-Yu; DE SCHEPPER, Koen; PAPADIMITRIOU, Panagiotis. Deep Multi-Agent Reinforcement Learning With Minimal Cross-Agent Communication for SFC Partitioning. **IEEE Access**, v. 11, p. 40384–40398, 2023. DOI: 10.1109/ACCESS.2023.3269576.

PHAM, Chuan; TRAN, Nguyen H.; REN, Shaolei; SAAD, Walid; HONG, Choong Seon. Traffic-Aware and Energy-Efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach. **IEEE Transactions on Services Computing**, v. 13, n. 1, p. 172–185, 2020. ISSN 19391374. DOI: 10.1109/TSC.2017.2671867.

QUICO, Franklin J. V.; BATTISTI, Anselmo L. E.; MUCHALUAT-SAADE, Débora; DELICATO, Flavia C. L-PRISM: A Domain-Specific Language for Describing Multimedia Service Function Chains. **Journal of the Brazilian Computer Society**, v. 31, n. 1, p. 545–569, ago. 2025. DOI: 10.5753/jbcs.2025.5453. Disponível em: <https://journals-sol.sbc.org.br/index.php/jbcs/article/view/5453>.

QUICO, Franklin Jordan Ventura; BATTISTI, Anselmo L. E.; MUCHALUAT-SAADE, Débora; DELICATO, Flavia C. A Domain-Specific Language for Multimedia Service Function Chains based on Virtualization of Sensors. In: PROCEEDINGS of the 30th Brazilian Symposium on Multimedia and the Web (WebMedia 2024). [S. l.]: Sociedade Brasileira de Computação - SBC, out. 2024. (WebMedia 2024), p. 11–19. DOI: 10.5753/webmedia.2024.243129.

RASMUSEN, E. **Games and Information: An Introduction to Game Theory**. 4. ed. Malden, MA; Oxford: Wiley, 2006. ISBN 9781405136662.

REHMAN, A. U.; AGUIAR, Rui. L.; BARRACA, João Paulo. Testing Virtual Network Functions Auto-Scaling using Open-Source Management and Orchestration. In: 2021 Telecoms Conference (ConfTELE). Leiria / Portugal: IEEE, 2021. P. 1–6. DOI: 10.1109/ConfTELE50222.2021.9435471.

REYHANIAN, Navid; FARMANBAR, Hamid; MOHAJER, Soheil; LUO, Zhi-Quan. 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). Virtual, Barcelona, Spain: IEEE, 2020. P. 4990–4994. DOI: 10.1109/ICASSP40776.2020.9054706.

RODIS, Panteleimon; PAPADIMITRIOU, Panagiotis. Intelligent and Resource-Conserving Service Function Chain (SFC) Embedding. **Journal of Network and Systems Management**, Springer Science e Business Media LLC, v. 31, n. 4, set. 2023. ISSN 1573-7705. DOI: 10.1007/s10922-023-09771-y.

ROSENTHAL, Robert W. A class of games possessing pure-strategy Nash equilibria. **International Journal of Game Theory**, v. 2, n. 1, p. 65–67, 1973. ISSN 1432-1270.

ROY, Sankardas; ELLIS, Charles; SHIVA, Sajjan; DASGUPTA, Dipankar; SHANDILYA, Vivek; WU, Qishi. A survey of game theory as applied to network security. In: IEEE. 2010 43RD HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES. Koloa, Kauai, Hawaii: IEEE, 2010. DOI: 10.1109/HICSS.2010.35.

RUIZ, Lidia; BARROSO, Ramón J. Durán; DE MIGUEL, Ignacio; MERAYO, Noemí; AGUADO, Juan Carlos; DE LA ROSA, Ramón; FERNÁNDEZ, Patricia; LORENZO, Rubén M.; ABRIL, Evaristo J. Genetic Algorithm for Holistic VNF-Mapping and Virtual Topology Design. **IEEE Access**, v. 8, p. 55893–55904, 2020. DOI: 10.1109/ACCESS.2020.2982018.

SANTOS, Guto Leoni; BEZERRA, Diego de Freitas; ROCHA, Élisson da Silva; FERREIRA, Leylane; MOREIRA, André Luis Cavalcanti; GONÇALVES, Glauco Estácio; MARQUEZINI, Maria Valéria; RECSE, Ákos; MEHTA, Amardeep; KELNER, Judith; SADOK, Djamel; ENDO, Patricia Takako. Service Function Chain Placement in Distributed Scenarios: A Systematic Review. **Journal of Network and Systems Management**, Springer US, v. 30, n. 1, p. 4, 2022. ISSN 15737705. DOI: doi.org/10.1007/s10922-021-09626-4.

SARRIGIANNIS, Ioannis; KARTSAKLI, Elli; RAMANTAS, Kostas; ANTONOPOULOS, Angelos; VERIKOUKIS, Christos. Application and Network VNF migration in a MEC-enabled 5G Architecture. In: 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). Barcelona / Spain: IEEE, 2018. P. 1–6. DOI: 10.1109/CAMAD.2018.8514943.

SBABOU, S.; SMAOUI, H.; ZIAD, A. **Equilibrium Analysis in Singleton Congestion Games**. Centre d'Économie et de Management de l'Océan Indien (CEMOI), Université de La Réunion, 2012. Accessed: 2025-07-28.

SCIANCALEPORE, Vincenzo; GIUST, Fabio; SAMDANIS, Konstantinos; YOUSAF, Zarrar. A double-tier MEC-NFV architecture: Design and optimisation. In: 2016 IEEE Conference on Standards for Communications and Networking (CSCN). Berlin / Germany: IEEE, 2016. DOI: 10.1109/CSCN.2016.7785157.

SHINDE, Swapnil Sadashiv; MARABISSI, Dania; TARCHI, Daniele. A network operator-biased approach for multi-service network function placement in a 5G network slicing architecture. **Computer Networks**, v. 201, p. 108598, 2021. ISSN 1389-1286. DOI: doi.org/10.1016/j.comnet.2021.108598.

SILVA, Thiago P.; BATISTA, Thais V.; BATTISTI, Anselmo L.; SARAIVA, Andre; ROCHA, Antonio A.; DELICATO, Flavia C.; BASTOS, Ian Vilar; MACEDO, Evandro L. C.; OLIVEIRA, Ana C. B. de; PIRES, Paulo F. A MAPE-K and Queueing Theory Approach for VNF Auto-scaling in Edge Computing. In: 2023 IEEE

12th International Conference on Cloud Networking (CloudNet). Hoboken, NJ, USA: IEEE, 2023. P. 144–152. DOI: 10.1109/CloudNet59005.2023.10490065.

SIMPY DEVELOPMENT TEAM. **SimPy: Discrete Event Simulation for Python**. [S. l.], 2025. Available at https://simpy.readthedocs.io/.

SON, Jungmin; BUYYA, Rajkumar. Latency-aware Virtualized Network Function provisioning for distributed edge clouds. **Journal of Systems and Software**, v. 152, p. 24–31, 2019. ISSN 0164-1212. DOI: doi.org/10.1016/j.jss.2019.02.030.

SONG, Enge; SONG, Yang; LU, Chengyun; PAN, Tian; ZHANG, Shaokai; LU, Jianyuan; ZHAO, Jiangu; WANG, Xining; WU, Xiaomin; GAO, Minglan; LI, Zongquan; FANG, Ziyang; LYU, Biao; ZHANG, Pengyu; WEN, Rong; YI, Li; ZONG, Zhigang; ZHU, Shunmin. Canal Mesh: A Cloud-Scale Sidecar-Free Multi-Tenant Service Mesh Architecture. In: PROCEEDINGS of the ACM SIGCOMM 2024 Conference. [S. l.]: ACM, ago. 2024. (ACM SIGCOMM '24), p. 860–875. DOI: 10.1145/3651890.3672221.

SOUALAH, Oussama; MECHTRI, Marouen; GHRIBI, Chaima; ZEGHLACHE, Djamal. A link failure recovery algorithm for Virtual Network Function chaining. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). Lisboa/Portugal: IEEE, 2017. P. 213–221. DOI: 10.23919/INM.2017.7987282.

SOUSA, Nathan F. Saraiva de; PEREZ, Danny A. Lachos; ROSA, Raphael V.; SANTOS, Mateus A.S.; ROTHENBERG, Christian Esteve. Network Service Orchestration: A survey. **Computer Communications**, v. 142-143, p. 69–94, 2019. ISSN 0140-3664. DOI: doi.org/10.1016/j.comcom.2019.04.008.

SUN, Gang; LI, Yayu; LIAO, Dan; CHANG, Victor. Service Function Chain Orchestration Across Multiple Domains: A Full Mesh Aggregation Approach. **IEEE Transactions on Network and Service Management**, v. 15, n. 3, p. 1175–1191, 2018. DOI: 10.1109/TNSM.2018.2861717.

SWENSON, Brian; MURRAY, Ryan; KAR, Soummya. On Best-Response Dynamics in Potential Games. **SIAM Journal on Control and Optimization**, v. 56, n. 4, p. 2734–2767, 2018. DOI: 10.1137/17M1139461.

TININI, Rodrigo Izidoro; SANTOS, Matias Romário Pinheiro dos; FIGUEIREDO, Gustavo Bittencourt; BATISTA, Daniel Macêdo. 5GPy: A SimPy-based simulator for performance evaluations in 5G hybrid Cloud-Fog RAN architectures. **Simulation Modelling Practice and Theory**, v. 101, 2020. ISSN 1569-190X. DOI: doi.org/10.1016/j.simpat.2019.102030.

TOSCA, OASIS. **TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0, Committee Specification Draft 04**. [S. l.]: Technical Report, 2017.

TOUMI, Nassima; BERNIER, Olivier; MEDDOUR, Djamal Eddine; KSENTINI, Adlen. Towards Cross-Domain Service Function Chain Orchestration. In: 2020 IEEE Global Communications Conference, GLOBECOM 2020 - Proceedings. Taipei, Taiwan: IEEE, 2020. DOI: 10.1109/GLOBECOM42002.2020.9348028.

TOUMI, Nassima; BERNIER, Olivier; MEDDOUR, Djamal-Eddine; KSENTINI, Adlen. On cross-domain Service Function Chain orchestration: An architectural framework. **Computer Networks**, v. 187, p. 107806, 2021. ISSN 1389-1286. DOI: doi.org/10.1016/j.comnet.2021.107806.

TOUMI, NASSIMA AND BAGAA, MILOUD AND KSENTINI, ADLEN. Hierarchical Multi-Agent Deep Reinforcement Learning for SFC Placement on Multiple Domains. In: 2021 IEEE 46TH CONFERENCE ON LOCAL COMPUTER NETWORKS (LCN). Edmonton, AB, Canada: IEEE, 2021. P. 299–304. DOI: 10.1109/LCN52139.2021.9524980.

VAQUERO, Luis M.; CUADRADO, Felix; ELKHATIB, Yehia; BERNAL-BERNABE, Jorge; SRIRAMA, Satish N.; ZHANI, Mohamed Faten. Research challenges in nextgen service orchestration. **Future Generation Computer Systems**, v. 90, p. 20–38, 2019. ISSN 0167739X. arXiv: 1806.00764.

VIEIRA, Juan Lucas; BATTISTI, Anselmo L. E.; MACEDO, Evandro L. C.; PIRES, Paulo F.; MUCHALUAT-SAADE, Débora C.; DELICATO, Flavia C.; OLIVEIRA, Ana C. B. Dynamic and Mobility-Aware VNF Placement in 5G-Edge Computing Environments. In: 2023 IEEE 9th International Conference on Network Softwarization (NetSoft). Madrid, Spain: IEEE, 2023. P. 53–61. DOI: 10.1109/NetSoft57336.2023.10175437.

VIEIRA, Juan Lucas; MACEDO, Evandro LC; BATTISTI, Anselmo LE; NOCE, Julia; PIRES, Paulo F; MUCHALUAT-SAADE, Débora C; OLIVEIRA, Ana CB; DELICATO, Flavia C. Mobility-aware SFC migration in dynamic 5G-Edge networks. **Computer Networks**, p. 110571, 2024. DOI: doi.org/10.1016/j.comnet.2024.110571.

VIOLA, Roberto; MARTÍN, Ángel; ZORRILLA, Mikel; MONTALBÁN, Jon; ANGUEIRA, Pablo; MUNTEAN, Gabriel-Miro. A Survey on Virtual Network Functions

for Media Streaming: Solutions and Future Challenges. **ACM Computing Surveys**, v. 55, n. 11, 2023. ISSN 0360-0300, 1557-7341. DOI: doi.org/10.1145/356782.

WANG, Shuyi; CAO, Haotong; YANG, Longxiang. A Survey of Service Function Chains Orchestration in Data Center Networks. In: 2020 IEEE GLOBECOM WORKSHOPS (GC WKSHPS. 2020 IEEE Globecom Workshops (GC Wkshps. Taipei, Taiwan: IEEE, 2020. P. 1–6. ISBN 978-1-72817-307-8. DOI: 10.1109/GCWkshps50303.2020.9367463.

XU, Qi; GAO, Deyun; LI, Taixin; ZHANG, Hongke. Low Latency Security Function Chain Embedding Across Multiple Domains. **IEEE Access**, v. 6, p. 14474–14484, 2018. DOI: 10.1109/ACCESS.2018.2791963.

XU, Yansen; KAFLE, Ved P. Optimal Service Function Chain Placement Modeling for Minimizing Setup and Operation Cost. In: IEEE CloudNet. [S. l.: s. n.], 2018. P. 1–3. DOI: 10.1109/CloudNet.2018.8549288.

XU, Zichuan; ZHANG, Zhiheng; LIANG, Weifa; XIA, Qiufen; RANA, Omer; WU, Guowei. QoS-Aware VNF Placement and Service Chaining for IoT Applications in Multi-Tier Mobile Edge Networks. **ACM Transactions on Sensor Networks**, 2020. ISSN 15504867. DOI: doi.org/10.1145/3387705.

YAGHOUBPOUR, Fatemeh; BAKHSHI, Bahador; SEIFI, Fateme. End-to-end delay guaranteed Service Function Chain deployment: A multi-level mapping approach. **Computer Communications**, v. 194, p. 433–445, 2022. ISSN 0140-3664. DOI: doi.org/10.1016/j.comcom.2022.08.005.

YI, Bo; WANG, Xingwei; LI, Keqin; DAS, Sajal k.; HUANG, Min. A comprehensive survey of Network Function Virtualization. **Computer Networks**, v. 133, p. 212–262, 2018. ISSN 1389-1286. DOI: doi.org/10.1016/j.comnet.2018.01.021.

ZAVLANOS, Michael; SPESIVTSEV, Leonid; PAPPAS, George J. A distributed auction algorithm for the assignment problem. In: PROCEEDINGS OF THE IEEE CONFERENCE ON DECISION e CONTROL (CDC). PROCEEDINGS of the IEEE Conference on Decision and Control (CDC). Cancun, Mexico: IEEE, 2008. P. 1212–1217. ISBN 9781424431243. DOI: 10.1109/CDC.2008.4739098.

ZHANG, Chuanji; JOSHI, Harshvardhan P.; RILEY, George F.; WRIGHT, Steven A. Towards a virtual network function research agenda: A systematic literature review of VNF design considerations. **Journal of Network and Computer Applications**, v. 146, p. 102417, 2019. ISSN 1084-8045. DOI: doi.org/10.1016/j.jnca.2019.102417.

ZHANG, Jianwei; ZHAO, Chenwei. Q-SR: An extensible optimization framework for segment routing. **Computer Networks**, v. 200, p. 108517, 2021. ISSN 1389-1286. DOI: doi.org/10.1016/j.comnet.2021.108517.

ZHANG, Tianzhu; QIU, Han; LINGUAGLOSSA, Leonardo; CERRONI, Walter; GIACCONE, Paolo. NFV Platforms: Taxonomy, Design Choices and Future Challenges. **IEEE Transactions on Network and Service Management**, v. 18, n. 1, p. 30–48, 2021. DOI: 10.1109/TNSM.2020.3045381.

ZHANG, Zhibo; WANG, Huiqiang; NIU, Shuangyue; LV, Hongwu. SP-ADMM: a distributed optimization method of SFC placement for 5G-MEC networks. In: THIRD International Conference on Algorithms, Microchips, and Network Applications (AMNA 2024). Xi' an, China: SPIE, 2024. v. 13171, p. 426–435. DOI: 10.1117/12.3031950.