National Technical University of Athens

School of Electrical and Computer Engineering

Division of Communication, Electronic and Information Engineering

Network Management and Optimal Design Laboratory

Optimal Resource Orchestration of Distributed 5G Applications over the Cloud Continuum

Dissertation submitted for the degree of Doctor of Philosophy

of

Ioannis Dimolitsas

Athens
November 23, 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering
Network Management and Optimal Design Laboratory

Optimal Resource Orchestration of Distributed 5G Applications over the Cloud Continuum

Dissertation submitted for the degree of Doctor of Philosophy of

Ioannis Dimolitsas

Advisory Committee: Symeon Papavassiliou
Theodora Varvarigou
Ioanna Roussaki

Approved by the seven-member committee on 23rd November 2023.

.....	.....	.....
S. Papavassiliou	T. Varvarigou	I. Roussaki
Professor NTUA	Professor NTUA	Assoc. Professor NTUA
.....	.....	.....
G. Matsopoulos	V. Karyotis	P. Papadimitriou
Professor NTUA	Assoc. Professor IU	Assoc. Professor UOM

.....
E. Stai
Assist. Professor NTUA

Athens, November 2023

.....
Δημολίτσας Ιωάννης
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

To my father, Nikos, in loving memory.

Abstract

The rapid advancement of fifth-generation (5G) wireless network technology has enabled a wide range of innovative applications with diverse requirements, including ultra-low latency, high reliability, and scalability, especially in the context of Internet-of-Things (IoT). To effectively support these applications, efficient resource management and orchestration are crucial. The Cloud Continuum (CC), which encompasses heterogeneous network infrastructures ranging from centralized cloud data centers to edge devices, offers a distributed computing and networking environment to meet the specific needs of 5G IoT applications deployment. However, orchestrating resources in this complex setting poses significant challenges. This thesis aims to address these challenges by proposing novel approaches for the management and orchestration (MANO) of computing and network resources and the CC layers, aligned with established frameworks like the European Telecommunications Standards Institute - Network Functions Virtualization (ETSI-NFV), in several aspects of the application's life-cycle, striving to strike a balance between the objectives of all the involved stakeholders.

Focusing on enabling the efficient deployment of emerging 5G services, optimized resource allocation, and enhanced overall application performance and Quality of Service (QoS), in this dissertation, the following challenges are tackled: (i) the infrastructure selection for network service deployment in the cloud continuum by taking into account the differences between infrastructure capabilities and the plethora of various individual requirements, (ii) the establishment of the Cross-Service Communication (CSC) and the Network Service Marketplace (NSM) to enable Virtual Network Function (VNF) sharing to further improve the resource utilization and enhance the QoS, (iii) distributed the virtual network embedding, to support modern IoT applications, which composed as interconnected VNFs, deployed distributed in the cloud continuum ecosystem, in the form of a hybrid Service Function Chain (hSFC), under strict resource capacity constraints, and (iv) design efficient and low-complexity algorithms to provide solutions to the corresponding problems, meeting the need for real-time decision making and re-optimization of such a dynamic environment.

In this context, initially, this work proposes an Edge Cloud (EC) Selection framework for network service deployment, focusing on network slices. It introduces a distributed service discovery framework for shared-enabled VNFs using a cache-based protocol, in order to effectively identify the demanded VNFs while eliminating the communication overhead. The framework, also, utilizes a Multi-Criteria Decision Making (MCDM) algorithm, specifically, the Analytical Hierarchy Process (AHP), to identify the most suitable edge infrastructure for Network Slice deployment in a low execu-

tion time, considering the functional, performance, and cost requirements of both the infrastructure provider and the user.

In addition, focusing on the QoS guarantees for IoT-based application deployment, where delay minimization plays a crucial role, a two-stage approach to provide distributed virtual network embedding (DVNE) solutions with minimized round-trip delay is introduced. Initially, a heuristic undertakes the association of VNFs of a Virtual Network Embedding (VNE) request; namely mapping, to a set of candidate ECs that can be hosted, while focusing on the minimization of the resource utilization within the EC infrastructure and increased co-location ratio between adjacent VNFs of the request. Given this initial mapping, an efficient path-based algorithm is proposed to construct the DVNE solution, in polynomial time. Based on graph theory, an augmented graph is constructed, on which several candidate paths for the DVNE are generated, using the k-shortest-paths algorithm. The embedding policy of the algorithm constructs optimal, or near-optimal solutions in polynomial time.

Moreover, to ensure the compatibility and interoperability of some of the proposed mechanisms with existing industry standards and practices, the integration of the proposed mechanisms and algorithms into a MANO architecture is considered, to support Industry 4.0 application deployments. Extending the standard ETSI NFV reference architecture, a deployment and orchestration mechanism for enabling CSC in industrial environments is proposed. For evaluation purposes, a Warehouse Robotics use case, which requires the collaboration of mobile robotic agents in an automated storage and retrieval system, is used.

Emphasizing the dynamic characteristics of CC, this thesis, finally, delves into the intricacies of dynamic resource allocation and autoscaling techniques. To this end, a multi-objective optimization mechanism for autoscaling has been developed. This mechanism leverages a multifaceted approach by integrating time series forecasting methods, specifically Auto-regressive Integrated Moving Average (ARIMA), to predict the incoming workload to applications' virtualized resources (containers) in the forthcoming time periods. Additionally, it incorporates the element of resource profiling per application, enabling the selection of the optimal application deployment, in terms of the number of identical replicas of containers. The primary objective of this mechanism is to strike a balance between the QoS guarantees for a given application, and, simultaneously, the mitigation of the administrative overhead for infrastructure providers by concurrently minimizing the overall energy consumption and resource utilization.

Keywords: Resource Orchestration, Multi-Criteria Decision-Making, Analytic Hierarchy Process, Edge Computing, Cloud Continuum, Virtual Network Embedding, Service Function Chain, Delay Minimization, Graph Theory, Internet of Things, Industry 4.0

Περίληψη στα Ελληνικά

Η ταχεία πρόοδος της τεχνολογίας ασύρματων δικτύων πέμπτης γενιάς (5G) έχει επιτρέψει την ανάπτυξη ενός ευρέος φάσματος καινοτόμων εφαρμογών, οι οποίες φέρουν ξεχωριστές απαιτήσεις που αφορούν την ποιότητα της υπηρεσίας (Quality of Service - QoS), όπως εξαιρετικά χαμηλή καθυστέρηση, υψηλή αξιοπιστία και ελαστικότητα, ειδικά υπό το πρίσμα του Διαδικτύου των Αντικειμένων (Internet of Things - IoT). Για την αποδοτικότερη υποστήριξη αυτών των εφαρμογών, η αποτελεσματική διαχείριση και ενορχήστρωση των υπολογιστικών και δικτυακών πόρων, είναι ιδιαίτερα σημαντική. Το Cloud Continuum (CC), το οποίο περιλαμβάνει ετερογενείς δικτυακές υποδομές που κυμαίνονται από κεντρικοποιημένες υποδομές σύννεφου έως κατανεμημένες στην άκρη του δικτύου, προσφέρει ένα κατανεμημένο περιβάλλον υπολογιστικών και δικτυακών πόρων για την κάλυψη των ειδικών χαρακτηριστικών ανάπτυξης εφαρμογών 5G-IoT. Ωστόσο, η ενορχήστρωση πόρων σε αυτό το πολύπλοκο περιβάλλον θέτει σημαντικές προκλήσεις. Αυτή η διατριβή στοχεύει να αντιμετωπίσει αυτές τις προκλήσεις προτείνοντας νέες προσεγγίσεις για τη διαχείριση και την ενορχήστρωση των υπολογιστικών και δικτυακών πόρων στα διάφορα επίπεδα του CC, ευθυγραμμισμένες με καθιερωμένα πρότυπα, όπως της Εικονικοποίησης Λειτουργιών Δικτύου (Network Function Virtualization - NFV).

Συγκεκριμένα, στη διατριβή αντιμετωπίζονται οι εξής προκλήσεις: (i) η επιλογή της κατάλληλης υποδομής για την ανάπτυξη υπηρεσιών δικτύου στο CC, λαμβάνοντας υπόψη την ετερογένεια των υποδομών και την πληθώρα των διαφόρων επιμέρους απαιτήσεων εφαρμογής, (ii) η εγκαθίδρυση αλληλεπίδρασης μεταξύ διαφορετικών υπηρεσιών που αναπτύσσονται ως δικτυακά τεμάχια μέσω κοινής χρήσης εικονικών συναρτήσεων δικτύου (Virtual Network Function – VNF), (iii) η κατανεμημένη ενσωμάτωση εικονικού δικτύου (Distributed Virtual Network Embedding - DVNE), για την υποστήριξη σύγχρονων εφαρμογών IoT που αναπτύσσονται ως διασυνδεδεμένα VNF, στη μορφή μιας υβριδικής αλυσίδας υπηρεσιών, υπό αυστηρούς περιορισμούς διαθεσιμότητας πόρων, (iv) η ανάπτυξη λύσεων για αυτόματη κλιμάκωση πόρων προς τη βελτιστοποίηση της διαχείρισης αυτών σε πραγματικό χρόνο, λαμβάνοντας υπόψη το απαιτούμενο φορτίο προς τα εικονικά μηχανήματα κάθε εφαρμογής, και (v) ο σχεδιασμός και η ανάπτυξη αποτελεσματικών και χαμηλής πολυπλοκότητας αλγορίθμων για να παρέχουν λύσεις στα αντίστοιχα προβλήματα, σε πραγματικό χρόνο, ώστε να ανταποκρίνονται στις δυναμικές συνθήκες του CC.

Σε αυτό το πλαίσιο, αρχικά, προτείνεται ένα πλαίσιο Επιλογής Υποδομής στα άκρα του Δικτύου (Edge Cloud - EC) για την ανάπτυξη εφαρμογών 5G, εστιάζοντας στην περίπτωση δικτυακών τεμαχίων. Προς αυτή την κατεύθυνση, αναπτύσσεται ένα Πλαίσιο Αναζήτησης Κατανεμημένων Υπηρεσιών για VNF, τα οποία υποστηρίζουν διατεμαχιακή επικοινωνία χρησιμοποιώντας ένα πρωτόκολλο που βασίζεται στην προσωρινή κρυφή μνήμη των επι-

μέρους ΕC, στοχεύοντας στην χαμηλή χρήση δικτυακών πόρων. Το πλαίσιο αυτό, επίσης, χρησιμοποιεί έναν πολυκριτηριακό αλγόριθμο λήψης αποφάσεων, για να προσδιορίσει την καταλληλότερη υποδομή ΕC για την ανάπτυξη δικτυακού τεμαχίου με χαμηλό χρόνο απόφασης, λαμβάνοντας υπόψη τη λειτουργικότητα, την απόδοση και απαιτήσεις κόστους τόσο του παρόχου της υποδομής όσο και του χρήστη της εφαρμογής.

Επιπλέον, η διατριβή προτείνει μια προσέγγιση δύο επιπέδων για την παροχή λύσεων DVNE με ελαχιστοποιημένη καθυστέρηση δικτυακής κίνησης μετ' επιστροφής. Ένας ευριστικός αλγόριθμος, αρχικά, αναλαμβάνει τη συσχέτιση των VNF ενός αιτήματος DVNE, σε ένα σύνολο υποψήφιων ΕC που αυτά θα μπορούν να ενσωματωθούν, εστιάζοντας στην ελαχιστοποίηση της χρήσης πόρων εντός της κάθε υποδομής ΕC. Δεδομένης αυτής της αρχικής χαρτογράφησης, προτείνεται ένας αποτελεσματικός αλγόριθμος βασισμένος σε λύσεις συντομότερων μονοπατιών, για την κατασκευή της λύσης DVNE, σε πολυωνυμικό χρόνο. Η πολιτική ενσωμάτωσης του αλγορίθμου κατασκευάζει βέλτιστες ή σχεδόν βέλτιστες λύσεις DVNE σε πολυωνυμικό χρόνο.

Ακόμη, για να εξασφαλιστεί η συμβατότητα και η διαλειτουργικότητα ορισμένων από τους προτεινόμενους μηχανισμούς με τα υπάρχοντα βιομηχανικά πρότυπα και πρακτικές, εξετάζεται η ενσωμάτωση των προτεινόμενων μηχανισμών και αλγορίθμων σε μια αρχιτεκτονική NFV, για την υποστήριξη της ανάπτυξης εφαρμογών βιομηχανίας 4.0 (Industry 4.0). Προτείνεται ένας μηχανισμός ανάπτυξης και ενορχήστρωσης για την ενεργοποίηση διατεμαχιακής επικοινωνίας σε βιομηχανικά περιβάλλοντα. Για λόγους αξιολόγησης, αναπτύσσεται ένα σενάριο χρήσης Warehouse Robotics, στο οποίο απαιτείται η συνεργασία κινητών ρομποτικών συσκευών σε ένα αυτοματοποιημένο σύστημα αποθήκευσης και ανάκτησης. Δίνοντας έμφαση στα δυναμικά χαρακτηριστικά του CC, στην παρούσα διατριβή μελετώνται και τεχνικές κατανομής πόρων και αυτόματης κλιμάκωσης (autoscaling). Υπό αυτό το πρίσμα, αναπτύχθηκε ένας μηχανισμός πολυκριτηριακής βελτιστοποίησης για autoscaling, ο οποίος συνδυάζει την πρόβλεψη του αναμενόμενου φορτίου μέσω μεθόδων πρόβλεψης χρονοσειρών, αλλά και την εκ των προτέρων χαρακτηρισμό πόρων ανά εφαρμογή, ώστε να επιλέξει τον αριθμό των αντιγράφων εικονικών μηχανημάτων (containers) σε περιβάλλον Kubernetes. Σκοπός του μηχανισμού είναι να συνδυάσει την εγγύηση ποιότητας υπηρεσίας μιας εφαρμογής, αντιστοιχώντας τη στη χρονική απόκριση, μειώνοντας παράλληλα το διαχειριστικό κόστος του παρόχου της υποδομής και τη συνολική κατανάλωση ενέργειας.

Οι προσεγγίσεις που προτείνονται αξιολογούνται αναλυτικά τόσο μέσω προσομοίωσης όσο και μέσω ενσωμάτωσής τους σε σύγχρονες λύσεις διαχείρισης και ενορχήστρωσης, ενώ παρέχονται συγκριτικά αποτελέσματα έναντι άλλων λύσεων από την υπάρχουσα πρόσφατη βιβλιογραφία. Τα συμπεράσματα που εξάγονται από τη θεωρητική και πειραματική αξιολόγηση των προτεινομένων πλαισίων συνοψίζονται στο καταληκτικό κεφάλαιο της διατριβής, όπου επιπλέον, αναπτύσσονται ανοιχτά ερευνητικά θέματα για μελλοντική εργασία.

Contents

Preface

Acknowledgements

Writing this thesis and revisiting the years of my Ph.D. journey, I realize that I would like to thank many people for their unwavering support.

Firstly, I would like to thank my advisor, Professor Symeon Papavassiliou, for his guidance and support throughout my Ph.D. studies. His trust in me and his valuable insights helped me the most to address the challenges that arose during my research journey. Additionally, I am thankful for the opportunity he gave me to join the NETMODE team and collaborate with many great colleagues.

Also, a big thank you goes to Dr. Dimitris Dechouniotis. The inspiration he shared with me regarding the research challenges we faced and his guidance played a crucial role throughout my Ph.D. journey. Of course, during this process, Dimitris was also there as a friend. His attitude in handling any issue that arose, his willingness to share my concerns beyond the confines of this thesis, and all the funny moments we had together have been invaluable over these years.

I would also like to thank all the excellent colleagues of NETMODE over the years and, especially, Dimitris and Maro, with whom we started this journey together, and of course, Marios, Kostas, Giorgos, and Vassilis. I am grateful that I met them, and after the many moments we shared, I can consider them my good friends.

I could not omit the valuable support provided to me by my friends Antonis, Spyros, Giannis, Theofilos, and Vassilis. The moments we spent together, the endless conversations we had about all sorts of topics, and their understanding of my concerns helped me the most all these past years.

Words are not enough to be able to express my gratitude to Dora. Her being by my side all these years, with her incomparable and multifaceted support and love, was truly priceless.

From the bottom of my heart, I would like to thank my family, and specifically, my sisters, Alexandra and Chrysanthi, my mother, Litsa, and my late father, Nikos. I thank them for their absolute trust in me, the sacrifices that they made, and their abundant support and love throughout my life.

Structure

The thesis is structured as follows:

Chapter 1 provides an introduction to the topics addressed in this thesis, discussing the research challenges and presenting the main contributions.

Chapter 2 covers the necessary mathematical background to understand the methods employed in addressing the identified problems. Additional information is provided within the relevant chapters when needed.

Subsequent chapters focus on specific problems deemed significant in this thesis. Each chapter begins by introducing the problem in its general context and reviewing relevant literature. The proposed framework is then presented, along with the modeling and analysis. The mathematical formulation of the problem and the proposed solution are discussed. Finally, a comprehensive evaluation of the framework is provided at the end of each chapter.

In **Chapter** 3, an Edge Cloud Selection framework for network service deployment, in the form of network slices, is proposed. In order to enable Cross-Service Communication and the establishment of the Network Service Marketplaces, a distributed service discovery framework for the shared-enabled Virtual Network Functions, which relies on a cache-based protocol is introduced. Also, a Multi-Criteria Decision Making algorithm that undertakes the identification of the most suitable edge infrastructure for Network Slice deployment, is exemplified. The method of the Analytical Hierarchy Process is exploited to provide the decision by taking into consideration both the infrastructure provider's and the user's functional, performance, and cost requirements. The individual mechanisms of the proposed architecture are evaluated, and compared with other state-of-the-art studies.

Considering the Network Function Virtualization technology, based on which a virtual service is referred to as a Virtual Network Function, and the requirements of modern 5G Internet of Things applications, in **Chapter** 4, a special case of the Virtual Network Embedding problem is approached; that is the distributed embedding of VNFs that compose a hybrid-Service Function Chain. Initially, a heuristic is developed, in order to undertake the optimal initial mapping of parts of a Virtual Network within an Edge Cloud infrastructure under various resource and quality of service constraints. Afterwards, the distributed VNE problem is formulated and modeled based on graph theory. The proposed shortest-path-based algorithm achieves optimal, or near-optimal solutions that minimize the round-trip delay in polynomial time. Furthermore, extended evaluation results are presented to highlight the efficiency of both algorithms, compared to relevant works in the literature.

In **Chapter** 5, an architecture that incorporates mechanisms developed to tackle the aforementioned resource orchestration problems is proposed. The discussed architecture aims to enable the deployment of industrial-specific services, as interconnected Virtual Network Functions in the form of Network Slices, and it is aligned with the recent NFV paradigm, alongside other established management and orchestration frameworks. The primary focus of the chapter revolves around three main aspects: the establishment of Cross-Service Communication, the selection of suitable

edge infrastructure, and the placement of adjacent Virtual Network Functions of a Network Slice. Furthermore, a comprehensive evaluation of the proposed approach is conducted using a real-world robotic collaboration scenario to assess its effectiveness and practical applicability.

Taking into consideration the importance of efficient dynamic resource provisioning in the context of Cloud Continuum, a multi-objective autoscaling framework for Edge Infrastructures is presented in **Chapter** 6. Specifically, resource profiles are defined, to provide a mapping between the quality of service and the computing resources. Furthermore, the Analytic Hierarchy Process is exploited to dictate the scaling decision of the resources under various Key Performance Indicators focusing on power optimization of the allocated resources. To guarantee maximum performance of the deployed applications, a time-series analysis model is dedicated to providing predictions regarding the incoming workload traffic. An exhaustive evaluation of the proposed framework is, also, provided to measure the benefit of the adopted techniques compared to state-of-the-art solutions.

Chapter 7 concludes the thesis by summarizing the key findings and observations derived from the research works presented earlier. We discuss the implications of our work and propose potential extensions and future directions for further research.

Chapter 1

Introduction

The number of devices connected to the Internet is growing exponentially, and this has created significant challenges for traditional cloud computing [1] as the amount of data generated by these devices increases. In the era of fifth-generation (5G) networks, the Internet of Things (IoT) [2] is a major contributor to this growth, as it comprises a vast network of sensors, smart devices, and other connected devices that generate data requiring processing and storage. It is predicted that by 2028, there will be an estimated 35 billion IoT connections [3]. These connections will include a variety of devices such as machines, sensors, wearables, connected cars, point-of-sale terminals, and consumer electronics [3], as shown in Fig. 1.1. At the end of 2023 there are expected to be around 3.3 billion Short-Range connected IoT devices. This number is projected to grow to 6 billion by 2028. The wide-area segment of these devices includes both cellular and low-power unlicensed technologies. By the end of 2028, it is anticipated that 60 percent of cellular IoT connections will be broadband IoT [3]. The traditional cloud computing model involves transferring this data to remote cloud data centers for processing and storage. However, this approach can lead to increased network congestion, latency, and security risks, particularly with the growth of IoT devices. The emergence of 5G networks provides an opportunity to meet this challenge by enabling faster and more reliable communication between devices, which will lead to significant advancements in various fields such as healthcare, transportation, and smart cities. However, the massive amount of data produced by these devices presents challenges related to data processing, storage, and transmission.

As demand for traditional cloud-based services such as video streaming, social networking, and online retail services continues to rise in mobile networks, there is also an increasing demand for new cloud services such as mobile cloud games, remote control services for vehicles, and services that manage manufacturing processes. However, meeting the increasing demand for computational requests in cellular networks has proved challenging for conventional cloud computing platforms that rely on data centers. These platforms have expanded their server capacities to improve the Quality of Service (QoS) they provide, but new computing platforms and architectures are needed to scale better with the explosion of mobile service requirements. Mobile traffic must pass through

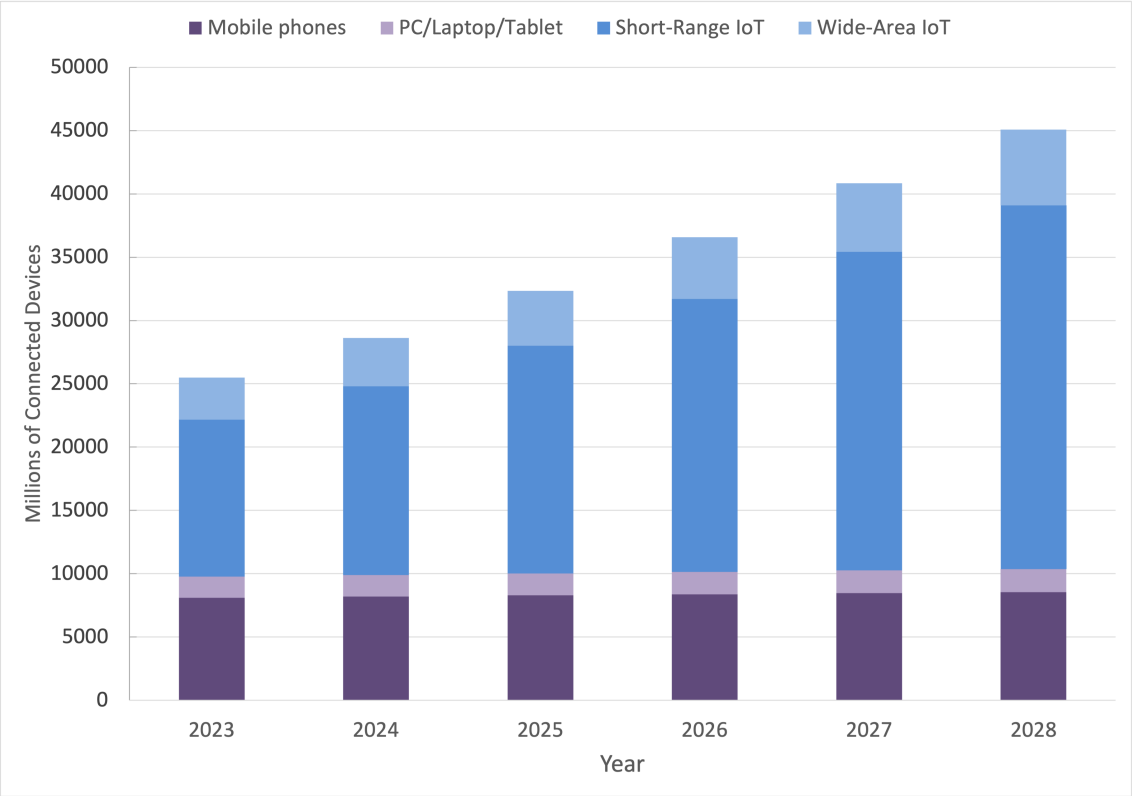


Figure 1.1: Connected Devices Forecast - Ericsson Mobility Report 2023 [3].

multiple stages, including the mobile backhaul and potentially the mobile core operator, for mobile users to access traditional cloud services. This process creates additional communication overhead that can exceed the latency requirements of critical services, while device mobility-related issues may occur [4, 5]. Additionally, intensive investments in computing and communication resources are required for conventional cloud computing platforms to enhance the quality of services provided in cellular networks. As a result, the costs of accessing cellular networks continue to increase.

Edge computing has been identified as a solution to overcome the challenges associated with cloud-based services in mobile networks. This approach involves processing data at the edge of the network, closer to where it is generated, instead of sending it to remote data centers [6]. Many works in the literature state that edge computing is a key enabler for many future 5G and beyond technologies such as IoT, augmented reality, and vehicle-to-vehicle communications by connecting cloud computing facilities and services to the end users [7]. The Edge computing paradigm provides low latency, mobility, and location awareness support to delay-sensitive applications. Architectural standardizations for realizing the capabilities of the edge computing paradigm have been proposed. Three well-known models are the Cloudlets [8], the Fog Computing [9], and Mobile Edge Computing [10].

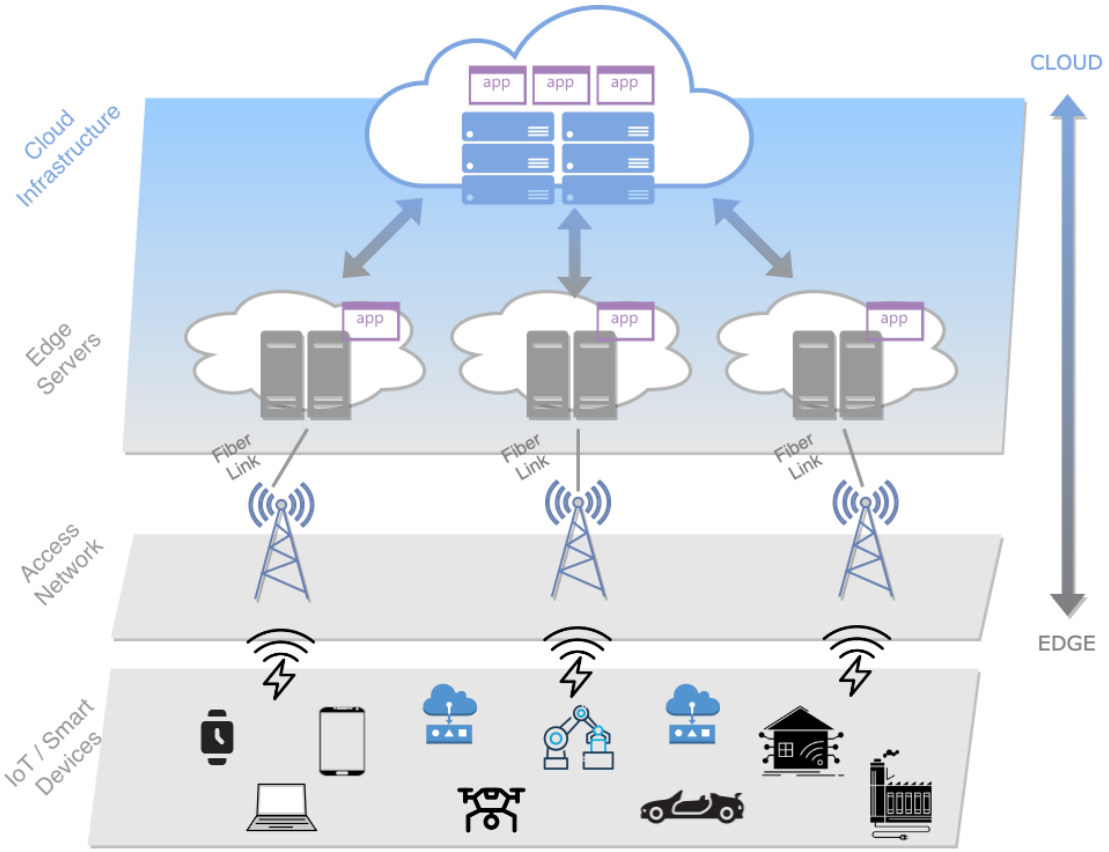


Figure 1.2: Mobile Edge Computing Architecture.

Mobile Edge Computing (MEC) is a standardized architecture provided by the European Telecommunications Standards Institute (ETSI), which enables computing, storage, and networking capabilities in the proximity of mobile users and devices, bringing computation and data storage closer to the end-users, improving service quality and reducing the burden on the core network. By deploying MEC servers in close proximity to base stations, mobile network operators can provide low-latency, high-bandwidth access to mobile users, enabling new applications and services such as augmented reality, real-time video analytics, and autonomous driving. Moreover, MEC enables service providers to move beyond traditional connectivity service models by leveraging their expertise in network management to bring cloud computing capabilities closer to the end-users. This allows for reduced latency, improved network efficiency, and more efficient use of network resources. With the evolution to cloud-native network functions and distributed cloud computing, service providers can further enhance their ability to offer adjacent industries new solutions that improve their operations and customer experiences.

Figure 1.2 illustrates the fundamental architectural aspects and key enablers of MEC. Starting

from the edge of the network, end devices, such as smartphones, tablets, and IoT sensors, generate or consume data, while communicating with base stations, which are wireless access points that provide connectivity between end devices and the network. In the context of MEC, base stations are equipped with additional computing and storage resources and are responsible for processing data locally before sending it to the core network or the cloud [11]. This fact reduces the latency and bandwidth requirements of the network. Above, edge servers are deployed in the proximity of the base stations and end devices. They are also responsible for hosting the edge applications, as well as the mandatory set of services and interfaces that manages the edge infrastructure resources, enabling the application deployment and the communication between all the involved edge/cloud components [12]. The decentralized *Management and Orchestration (MANO)* of the edge resources are performed by a MEC platform, which is typically located between the edge servers and the cloud infrastructure in a MEC architecture. It enables interoperability between the two and provides a platform for deploying and managing applications and services at the network edge [13]. The MEC platform can be implemented in different ways, such as a software layer on top of the edge servers or as a standalone platform that aggregates resources from multiple edge servers, in a centralized, hierarchical manner [14]. Furthermore, enables the development of context-aware and immersive applications, including a set of management and orchestration functions to manage the lifecycle of the applications running on the edge servers [15] and ensure their efficient operation. On top, the centralized cloud infrastructure provides cloud services, such as storage, processing, and monitoring. In the context of MEC, the cloud infrastructure is, also, used to provide collaborating task-offloading capabilities alongside the edge servers [16, 17]. The cloud infrastructure can also perform centralized management and orchestration operations, support to distributed services of the MEC platform, and global optimization capabilities for several objectives, such as end-to-end latency minimization, power consumption mitigation, and efficient resource selection.

Virtualization is a key technology that enables the capabilities of MEC [18]. By using virtualization techniques, the computing resources of cloud/edge infrastructures can be shared among multiple virtual machines, allowing different applications to be hosted on the same physical hardware. This enables efficient use of resources, reduces hardware costs, and enables greater flexibility in managing and deploying applications. Two technologies that play an important role in virtualization, especially in the cloud and MEC context, are Network Function Virtualization (NFV) introduced by ETSI [19] and Software-Defined Networking (SDN) [20]. NFV enables the deployment of network functions, such as firewalls, routers, and load balancers, on virtual machines running on edge servers, instead of on dedicated hardware devices. This allows service providers to deploy and manage network functions more flexibly and efficiently [21, 22]. SDN, on the other hand, enables centralized control of network traffic flows by separating the control plane from the data plane. Utilizing NFV and SDN enhances flexibility regarding managing and controlling computing, network resources, and traffic flows.

Leveraging the virtualization technologies and the MEC capabilities, *Network Slicing* is a technology that enables service providers to divide a single physical network into multiple virtual net-

works, each with its characteristics and functions, such as specific quality of service, bandwidth, and security. By utilizing network slicing, service providers can offer customized and innovative services that meet the unique requirements of different market segments and expand their business opportunities. In this way, network slicing plays a crucial role in enabling service providers to enter new markets and introduce new services.

5G verticals in MEC environments could benefit from the adoption of network slicing to create isolated network environments for different tenants. While this allows for improved isolation and security, it also presents an opportunity for Cross-Service Communication (CSC) within the same Edge Cloud (EC) infrastructure [14, 23]. This means that network services provided by one tenant can consume the network services of another tenant. CSC can enhance the overall performance in terms of throughput, latency, and other key performance indicators [24]. It also opens up opportunities for financial rewards through service leasing. Additionally, service components can be shared among different service operators, enabling inter-service operability [25, 26]. These emerging trends are paving the way for the development of a next-generation Network Service Marketplace (NSM), where end-users can compose custom applications and services can consume other co-located services in a controlled and secure manner [27, 28]. The enabling of NSM brings additional orchestration-related requirements in the context of MEC, such as the demand for automated functionalities for service providers and consumers, and support of the entire life cycle of network services including registration, discovery, instantiation, and termination [29]. An NSM must provide essential mechanisms for service registration and discovery to allow users to specify the features and requirements of their network services in a consistent and open manner that can be used by the participating heterogeneous MEC infrastructures. Of course, existing data models, such as the ETSI NFV data model, propose a specific analytical description of the structure and parameters of network services towards the automated deployment and orchestration of virtualized 5G applications.

Furthermore, in the context of MEC and 5G, applications can also be deployed as service composition through virtualization and orchestration techniques. For example, containerization is a virtualization method of packaging an application and its dependencies into a lightweight, standalone executable that can be run consistently across different computing environments [30]. Orchestration refers to the automated management of containers and their underlying resources to ensure that the applications are deployed and run efficiently and effectively. Service deployment in MEC requires consideration of factors such as network latency, processing power, storage capacity, and energy consumption, among others. As a result, service providers need to carefully design and manage their service deployment strategies to ensure that they are optimized for the specific requirements of each application and its end-users.

In light of the distinctive features offered by cloud computing and MEC architectures, coupled with the proliferation of data generation and processing from end-user, or IoT devices, as well as advancements in technologies like virtualization, there arises a compelling demand for optimizing resource utilization across the entire computational layers of the network. This includes the di-

verse infrastructures and functionalities that serve modern applications, spanning from centralized cloud resources to edge computing nodes and smart devices at the network's edge. Encompassing the integration of resources and services across the central network, its edge, and in-transit locations along the data path contributes to the emergence of one of the most recent paradigms discussed in the cloud computing domain known as the *Cloud Continuum*[31]. This continuum paradigm facilitates the integration of diverse, heterogeneous infrastructures, spanning from cloud environments to mobile devices, and encompassing intermediary components such as Internet Service Provider (ISP) gateways, cellular base stations, and private cloud deployments [32]. So, the Cloud Continuum can be considered as a concept that refers to the seamless integration of various edge/cloud resources into a single, flexible computing environment. Other, emerging factors promoting this computing model are the adoption of microservices architecture, as well as the Function-as-a-Service development model [33]. Based on [34], the ***Cloud Continuum (CC)*** *can be defined as a computing paradigm that extends the traditional cloud, formatting a continuum pool of resources from the network edge to the core, providing analysis, processing, storage, and data generation capabilities.*

1.1 Challenges and Motivation

As mentioned, the combination of MEC and 5G technology holds great potential for providing innovative and efficient services to end-users, especially performing the construction of the unified computational model of the CC. However, the realization of this potential requires addressing several challenges. Micro Data Centers, also referred to as EC infrastructures, are distributed across diverse geographic locations and come in varying scales and types with different computing resources, such as processing, storage, and network resources. This heterogeneity of the available resources aligns with the objective nature of the requirements of the different applications to be supported. Furthermore, the subjective demands of 5G verticals for ultra-reliable and low-latency communication (URLLC) for service delivery, along with the management and orchestration of EC resources and guaranteeing the applications' requirements, highlight several research issues that concern both the academic and industrial communities. As a result, complex optimization problems need to be modeled to capture dynamic changes in the context of 5G verticals applications, especially those that require low latency and support for computational offloading. The limited availability of EC resources, coupled with the increased demand arising from the continuous growth of connected devices, exacerbates these challenges, thus underscoring the need for further research and development in this area. This thesis aims to propose and develop solutions for the open research challenges identified in the realm of MEC, 5G, and Cloud Continuum, in general. The main goal is to address the issues related to the orchestration and management of MEC resources, while exploiting available cloud resources, as well as the design of efficient algorithms to optimize the performance of continuum systems, considering the heterogeneity of the available resources and the diverse requirements of the different applications that will be supported, towards the exploitation

of the full potential of cloud continuum paradigm:

- **Resource Selection at the Network Edge**: Edge Cloud resource selection refers to the level of automation with respect to the selection of hardware and software resources. It is a challenging task in the context of edge computing, and it entails identifying the optimal EC resources for hosting virtualized applications based on various factors, including application characteristics, resource availability, network conditions, and connectivity constraints [35]. In practice, it is a service embedding problem where application components must be placed as a composition in a specific Point of Presence for deployment [24]. To achieve this, edge computing platforms must employ sophisticated algorithms and models for EC selection, monitor application performance in real-time, and dynamically adjust infrastructure selection when needed to ensure optimal performance and efficiency. The primary requirements of EC selection include low execution time and the multi-objective nature of the problem. Low execution time is critical as EC selection decisions must be made quickly to ensure that virtualized applications are deployed on the optimal edge cloud infrastructure in a timely manner. Thus, single-objective decision methods would not select an optimal solution. On the other hand, many multi-criteria decision-making (MCDM) approaches have been proposed for the cloud provider selection problem [36] for the deployment of cloud services. These methodologies take into account various types of Key Performance Indicators (KPIs) and rank the candidate EC providers. Furthermore, they enable the service owners to adjust the individual weight of every criterion based on their hard and soft requirements. MCDM optimization techniques are also essential for balancing competing objectives such as application performance, resource utilization, and cost, and for finding optimal solutions by considering all relevant objectives simultaneously [37].
- **Network Service Marketplace and Cross Service Communication establishment**: The deployment of 5G verticals in ECs is, also, based on the concepts of network slicing and network services, which provide the desired isolation among tenants. However, the co-hosting of such service ecosystems creates opportunities for cross-service interactions within the same EC infrastructure, where a Network Service (NS) can consume an NS of a different tenant. The CSC enhances the overall performance in terms of throughput, latency, and potentially more KPIs, and creates opportunities for financial rewards through service leasing [23]. In a similar manner, service components may be shared among different service operators, i.e., a service orchestration aspect, known as inter-service operability [38]. As mentioned earlier, these emerging trends pave the way towards next-generation NSMs, where users are able to compose custom service chains, while a service will be empowered to consume other co-located services in a controlled and secure manner [39]. The NSM should provide several automated functionalities to both service providers and consumers, while supporting the entire life cycle of the network services, including service registration, discovery, instantiation, and termination. Furthermore, it must provide secure and automated mechanisms for CSC establishment. Especially, with the increasing number of services and devices at the CC, the

challenge of service discovery becomes more critical for efficient orchestration [40]. Service discovery mechanisms should be able to locate services efficiently, even in highly dynamic, distributed, and heterogeneous environments. Furthermore, service discovery should be able to take into account factors such as service quality, network latency, and service availability.

- **Latency-aware service deployment and configuration**: The performance and user experience of services at the edge of the network are highly dependent on the deployment and configuration of the underlying infrastructure. The challenge of delay-aware service deployment and configuration involves developing mechanisms to optimize the deployment and configuration of services in real-time, taking into account factors such as network latency, available resources, and service requirements. This challenge is particularly critical for services that require low latency, such as smart city applications [7], remote healthcare [41], and autonomous driving [42]. Especially, in the context of CC, deployments require to be distributed across multiple domains. This problem is referred to as Virtual Network Embedding (VNE) and aims at determining the placement of virtual services both internally in the edge/cloud infrastructure and distributed, among distinct ECs, under several resource and network constraints [43]. With regard to NFV technology, a virtual service is referred to as a Virtual Network Function (VNF). In this context, many resource orchestration platforms have been developed that rely on either virtual machines (VM) or containers, such as OpenStack [44] and Kubernetes [45], and they provide several functionalities about service embedding, resource allocation, and networking. A Service Chain Function (SFC) includes several VNFs with a predefined execution order and specific computing and network requirements [46]. The typical VNE problem focuses on embedding the SFC on the substrate network. As edge computing provides geographically spread small-scale resources, the delay minimization in the placement of VNFs becomes more challenging. Furthermore, in IoT applications, which usually include online machine learning processing [47], the forward traffic consists of raw data for processing at the edge or cloud, while the backward traffic includes results and models that must be sent back to the end devices that generate the network traffic. This possibly requires different forward and backward paths for the SFC. In the literature, various studies discuss VNE solutions both in a single domain or distributively, aiming at minimizing the end-to-end network delay. However, to meet the strict requirements of emerging IoT applications, innovative research should be directed to deal with round-trip delay minimization [48].

- **Service Reliability and Resource Scalability**: Service reliability is a critical factor in ensuring the quality of end-user experience [18]. EC infrastructures are subject to failures and disruptions, which can result in service downtime and data loss. The challenge of service reliability and resilience involves developing mechanisms to ensure that services remain available and functional, even in the face of failures or disruptions. This challenge includes developing mechanisms for resource autoscaling, load balancing, and service placement re-

optimization. As mentioned, the availability of computing resources, especially at the network edge is subject to significant variations due to changing demand and resource utilization. The management of resources for IoT-based applications presents a significant challenge, and numerous orchestration platforms, leveraging virtualization technology, offer comprehensive operations to handle the complete life-cycle of such applications. Notably, OpenStack [44] serves as a widely adopted Virtualized Infrastructure Manager (VIM) for the effective management of VMs in cloud infrastructures. Similarly, Kubernetes [45] is extensively used for orchestrating and managing containerized applications. These platforms enable automated deployment, scaling, management, and maintenance of applications. In terms of autoscaling, both OpenStack and Kubernetes support horizontal and vertical scaling. Horizontal scaling is commonly employed across both platforms, whereas vertical scaling involves restarting the VM (or container) and is therefore unsuitable for real-time resource management. These scaling mechanisms exhibit a certain level of granularity, primarily relying on basic monitoring metrics like CPU and memory utilization while overlooking other critical performance parameters such as dynamic workload and energy consumption. To meet the stringent requirements of emerging 5G applications, including stringent demands for delay, throughput, and location, there is a pressing need to enhance existing scaling mechanisms to address the underlying challenges more effectively.

- **Energy Consumption Mitigation**: The proliferation of IoT devices and the exponential growth of data they generate are putting alongside with the increasing pressure on energy consumption. Extensive research is needed to understand the energy profiling characteristics of devices, infrastructures, and applications. To achieve energy-efficient orchestration of IoT applications, it is crucial to study and exploit trade-offs between performance and energy savings [49]. In some cases, sacrificing a certain level of performance may be acceptable in exchange for significant energy savings. By integrating green computing principles into the CC orchestration, it is possible to effectively manage the environmental impact of data traffic growth while harnessing the full potential of 5G Virtualization, and MEC technologies for innovative and efficient service delivery in the CC. Orchestrating resources in a power-efficient way is essential to enable Green Computing resource management solutions [50]. The development of dynamic resource provisioning techniques could lead to the minimization of power consumption and simultaneously guarantee high QoS in line with the workload demand. The challenge of energy-efficient orchestration involves developing mechanisms to optimize the energy consumption of EC infrastructures while ensuring that services remain available and responsive [51]. Energy-efficient orchestration mechanisms should be able to dynamically adjust the allocation of computing resources and optimize the scheduling of services to minimize energy consumption. Achieving optimal energy consumption in Cloud Continuum ecosystems requires developing and linking appropriate energy consumption models for all components of the system. This type of research entails significant effort but can lead to effective solutions that balance performance and energy efficiency, ultimately contributing

to sustainable and energy-conscious IoT systems.

1.2 Contributions

The research and development efforts in this thesis are dedicated to addressing the multifaceted, above described challenges and problems associated with the infrastructure and application orchestration in the context of Cloud Continuum. These challenges encompass various aspects of the cloud continuum ecosystem, concerning several stages of the life-cycle management of distributed 5G applications, taking into account multiple KPIs that refer to the infrastructure provider, service provider, and application objectives. Through innovative solutions and methodologies, this thesis aims to tackle these challenges and issues to ensure effective application life-cycle management and resource orchestration. This includes the deployment, configuration, operation, and maintenance of applications at the network edge, with a focus on optimizing performance, and reliability. By addressing these aspects, this research endeavors to contribute to the seamless integration of edge/cloud resources with the broader 5G and MEC ecosystem, and enable the realization of the full potential of Cloud Continuum for next-generation applications and services. The contributions to the above topics can be summarized in the following:

- **Multi-Objective Edge Cloud Selection**: The advancement of 5G network service ecosystems has created a need for efficient service discovery among multiple ECs that may have diverse resources and features. In addition to facilitating 5G service deployment, such a service discovery architecture can enable CSC, allowing one NS to consume another NS and enhance its own service capabilities. In this thesis, a distributed service discovery mechanism for network service marketplaces in edge computing is introduced. To deal with the need for distributed solutions that minimize the network overhead, the proposed mechanism utilizes a cache-based protocol for the discovery of network services, which are enabled for sharing, among geographically spread ECs. The cache-based discovery enables distributed query messages propagation regarding VNFs which are available for CSC. The adoption of this message-forwarding policy aims to eliminate the communication overhead over the EC network, where multiple end-devices and micro-datacenters interact to support the emerging 5G verticals in the CC. Furthermore, a multi-criteria ranking mechanism is introduced, in order to identify the most suitable EC based on the application's functional, performance, and cost requirements. To meet the dynamically changes conditions where time constraints for IoT application deployment are strict, the Analytic Hierarchy Process (AHP) [24] is used as the core algorithm of the multi-criteria ranking mechanism. AHP offers several advantages for multi-objective decision-making in the context of orchestration mechanisms. It provides a simple and efficient approach to decision-making with relatively low computational complexity. This makes it suitable for integration into orchestration mechanisms that require timely decision-making processes. Additionally, can be flexibly customized in terms of the criteria that are considered in the process, as well as the number of alternatives under evaluation.

This scalability makes it suitable for complex decision-making scenarios in the orchestration of network services, where multiple criteria and alternatives need to be considered simultaneously.

- **Resource Mapping of Virtual Network Functions**: Effective resource allocation for virtual network embedding is a critical aspect of network virtualization and plays a crucial role in optimizing resource utilization, enhancing performance, and ensuring efficient service provisioning. A typical version of the VNE problem is the allocation of resources within an EC infrastructure [52]. In essence, this problem involves the mapping of virtual network requests onto physical resources, such as servers, links, and switches, in a way that meets the performance and resource requirements of the corresponding application. As mentioned above, NFV's scope has expanded beyond telecommunications to include cellular networks [5], [6], as well as other domains such as automotive, media, e-health, and manufacturing, known as verticals [7]. In an NSM ecosystem, the aspect of cross-service interactions, which could enhance the quality of service delivery at the network edge [53], emerged as a critical constraint of the VNE problem. In this dissertation, a formulation of this problem is proposed taking into consideration the CSC requirement between VNFs of distinct service chains. Under the infrastructure's capacity constraints, a heuristic undertakes the mapping of VNFs that have to be deployed based on their resource demands, and, simultaneously, a partitioning of the VNE request is performed, based on the sequence of VNFs of the request. The aim is to provide an efficient mapping between the partitions of the Virtual Network (VN), in terms of resource utilization, and effective VNF placement within the infrastructure. That mapping refers both to the VNF embedding on the underlying physical servers and the virtual links between them onto the physical network links between the infrastructure's servers and network switches. The introduced heuristic leverages the CSC requirement to perform the mapping, as some VNFs of the request are already deployed on specific servers of the corresponding EC. In addition, a combination of the best-fit and worst-fit approximation methodologies of the bin-packing problem [54], leads to low execution time, and a high co-location ratio between adjacent VNFs of the request, which is an important factor regarding the efficient resource utilization and service deployment in the context of NFV and network slicing [55].

- **Distributed Virtual Network Embedding for Delay Minimization**: Distributed Virtual Network Embedding (DVNE) plays a crucial role in addressing the emerging challenges of IoT-based 5G applications management and resource orchestration. With the rapid growth of edge computing capabilities and the increasing demand for latency-sensitive applications, there is a need to efficiently allocate and manage resources in a distributed manner, allowing for the seamless integration of diverse services and applications. By distributing the network embedding process, DVNE improves scalability, flexibility, and resource utilization in edge environments. It enables the allocation of network functions and resources closer to the edge,

reducing latency and enhancing the overall performance of IoT applications. This dissertation, therefore, focuses on the modeling of the DVNE problem, by considering the dynamic aspects of the IoT applications deployment in the CC, aiming to minimize the round-trip network delay. Latency minimization in VNE has been approached in the literature, leveraging graph theory [56, 57], and such modeling is, also, used in this thesis regarding the proposed DVNE formulation. The round-trip delay is the dominant performance metric in edge computing systems, especially for QoS guarantee of real-time applications related to IoT [42, 58]. Minimizing the round-trip delay ensures that the applications meet their performance requirements and provide a satisfactory user experience, as the limited computing capabilities of IoT devices require a lot of tasks to be performed fully on edge computing infrastructures or partially via computational offloading. Furthermore, for IoT applications, which usually include online machine learning processing [41], the forward traffic consists of raw data, while the backward traffic includes results and models that must be sent back to the end devices that generate the network traffic. It is worth mentioning that in the dynamic environment, where such applications operate, conditions frequently change due to end-device mobility and other orchestration constraints related to the limited edge resources, which could lead to the need for recurrent re-embedding of the VNFs of several SFCs, to obtain optimal, or near-optimal solutions. Thus, a shortest-path-based algorithm that identifies the DVNE solution that minimizes the round-trip delay is proposed. The proposed solution utilizes a heuristic to perform an initial mapping of the VNFs for each VNE request among the distributed EC infrastructures to meet a set of resource utilization constraints. Also, in the formulation, it is assumed that the VNFs that compose the VNE request could be already deployed on specific ECs, or are required to be deployed from scratch, with the respective compute and network resource requirements. Afterward, a path-based algorithm constructs a DVNE solution that minimizes the round-trip network delay, by calculating candidate embedding paths; that is paths in the substrate network graph model. The k-shortest paths algorithm by Yen [59] is utilized. An extended evaluation of the proposed approach is given, and shows its benefits regarding the low complexity and the approximation to the optimal solution, compared to existing techniques.

- **Impact of Optimized 5G Applications Orchestration Solutions in Industrial IoT**: Efficient resource orchestration plays a crucial role in maximizing the potential benefits of 5G networks and edge computing, particularly in the context of technological advancements like Industry 4.0 [60]. In the context of Industry 4.0, which focuses on integrating digital technologies into industrial processes, resource orchestration becomes even more critical. The implementation of 5G networks and edge computing enables the deployment of advanced technologies such as IoT devices, real-time data analytics, and intelligent automation, all of which are essential for Industry 4.0 applications [61]. Thus, the aforementioned open issues regarding resource orchestration, Edge Cloud Selection, CSC establishment, and Virtual Network Embedding, also determine the potential benefits that can be derived from the

establishment of 5G networks and edge computing in technological areas such as Industry 4.0 [62]. These benefits include improved operational efficiency, enhanced automation and control, real-time data-driven decision-making, predictive maintenance, and the ability to develop innovative products and services. Toward this direction, in this thesis, a scenario of collaborative warehouse robotics application is implemented to highlight the benefits of mechanisms regarding the edge infrastructure selection, the VNF placement, as a VNE type problem, and the effectiveness of CSC, where the collaboration of different services is necessary. In precise, an ETSI NFV-aligned architecture is implemented to enable the deployment of industrial-specific interconnected VNFs as network slices, at the edge of the network. Furthermore, in this architecture, mechanisms that facilitate CSC are included allowing network slices of different owners to communicate towards the completion of collaborative tasks of an industrial production process. As mentioned, the proposed benefits of the optimized orchestration are evaluated through a use case in the area of Warehouse Robotics services [63], which requires the coordination of mobile robotic agents of different vendors on the factory floor. The evaluation focuses on the low-latency interaction and the mission's completion time.

- **Multi-Objective Resource Autoscaling in Edge Computing Infrastructures**: In order to provide a comprehensive solution regarding resource orchestration in the context of CC, with an efficient application life-cycle management, the dynamic autoscaling of resources emerges as a critical component. The integration of autoscaling capabilities in resource orchestration solutions enables dynamic, real-time adjustments of the allocated resources that support an application, depending on workload fluctuations and changing conditions in the edge/cloud computing and network infrastructures [64]. The development of dynamic resource provisioning techniques holds the promise of not only minimizing power consumption and making green computing resource management solutions feasible, but also ensuring a high level of service quality that aligns seamlessly with the prevailing workload demands. Therefore, this dissertation focuses on the modeling of the dynamic resource autoscaling problem, by taking into account the dynamic aspects of the IoT applications deployment in the CC, aiming to address the challenge of balancing between the QoS guarantees and the minimization of the power consumption and the resource utilization. Toward this direction a multi-objective autoscaling framework aligned with the Kubernetes architecture and state-of-the-art practices is proposed, to dictate the scaling decision process by incorporating competing criteria, specifically aiming at minimizing power consumption and efficient resource utilization, by exploiting the AHP algorithm. A key contribution lies in the formulation of resource profiles for each application, establishing a linkage between the application's QoS requirements and the allocation of computing resources, leading to the maximization of system performance. The exploitation of the resource profiles in combination with the lightweight hierarchical decision-making algorithm of the AHP, enables timely real-time scaling decisions of the resources in edge computing infrastructures, meeting the requirement for fast

adaptations in the context of CC. In addition, to combine the benefits of proactive scaling approaches with the proposed framework, an Autoregressive Integrated Moving Average (ARIMA) [65] model is employed to perform highly accurate workload traffic predictions for the corresponding applications that are deployed in an infrastructure. Also, the proposed framework is evaluated against a realistic dataset in a small-scale testbed. Numerical results indicate that the combined reactive and proactive nature of the proposed framework leads to a significant reduction of the average energy consumption and resource utilization when compared to other state-of-the-art techniques.

At the next chapter, the necessary mathematical background to understand the methods employed in addressing the identified problems, is discussed.

Chapter 2

Background

This chapter aims to present the fundamental background required for comprehending the methods employed in the subsequent sections of this work. Any supplementary information necessary will be supplied in the main body of the thesis as and when it becomes relevant.

2.1 Analytic Hierarchy Process

The Analytic Hierarchy Process is a multi-criteria decision-making technique [66] employed to prioritize and assess alternatives by considering a predefined set of criteria. AHP is widely used in various cases, such as product design and operational research [67]. AHP is designed to accommodate diverse types of KPIs that may exhibit distinct data characteristics. This method utilizes a hierarchical structure that consists of (i) KPIs of several types and (ii) attributes, which summarize a group of KPIs. For each KPI, a value v is determined for every competing alternative. In addition, the importance of each criterion q is defined, as a weight value w_q. The weight value indicates the significance of the lower-level attribute or KPI in determining the value of the upper-level attribute [68]. We assume that the weight of the attribute or KPI j at level i, $w_{ij} \leq 1$, and the sum of weights of a group of siblings attributes or KPIs of an attribute j at level i, $\sum w_{ij}$ is equal to 1. Here, we delve into the basic calculations regarding AHP for handling different types of KPIs and provide an evaluation of a set of alternatives.

2.1.1 Relative Comparison Matrix Calculations.

Considering a set of N alternatives, for each KPI X a Relative Comparison Matrix (RCM), which reflects the AHP priority vector, as this is defined in [69], is computed to represent the pairwise comparison between the competing alternatives for the respective KPI. We assume that u_i and u_j are the values for the KPI X that correspond to the alternatives A_i and A_j. Then, the RCM is calculated as the ratios between pairs of alternatives as follows:

$$RCM_X = \begin{bmatrix} 1 & A_1/A_2 & \dots & A_1/A_N \\ A_2/A_1 & 1 & \dots & A_2/A_N \\ \vdots & \vdots & & \vdots \\ A_N/A_1 & A_N/A_2 & \dots & 1 \end{bmatrix}. \tag{2.1}$$

It is worth mentioning that for various types of KPIs' values, the pairwise comparison is based on different calculation methods. For four main numerical types, the calculations are shown in Table 2.1. The AHP could also handle linguistic (fuzzy) values to perform the evaluation of the alternatives. That AHP version is named Fuzzy Analytic Hierarchy Process (FAHP) [70].

Table 2.1: Relative ranking model for the four types of numerical values [71]

Numeric KPI: $A_i/A_j = \begin{cases} v_i/v_j & \text{if higher is better} \\ v_j/v_i & \text{if lower is better} \\ w_q & \text{if } \nexists v_i \\ 1/w_q & \text{if } \nexists v_j \end{cases}$	Boolean KPI: $A_i/A_j = \begin{cases} 1 & \text{if } v_i = v_j \\ w_q & \text{if } v_i = 1 \wedge \\ & \quad \wedge v_j = 0 \\ 1/w_q & \text{if } v_i = 0 \wedge \\ & \quad \wedge v_j = 2 \end{cases}$
Unordered Set KPI: For essential attributes $A_i/A_j = \frac{size(v_i)}{size(v_j)}$ For non-essential attributes $A_i/A_j = \begin{cases} \frac{len(v_i \cap v_r)}{len(v_i \cap v_r)} & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \\ 1 & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ w_q & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ 1/w_q & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \end{cases}$	Range KPI: For essential attributes $A_i/A_j = \frac{len(v_i \cap v_r)}{len(v_i \cap v_r)}$ For non-essential attributes $A_i/A_j = \begin{cases} \frac{len(v_i \cap v_r)}{len(v_i \cap v_r)} & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \\ 1 & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ w_q & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ 1/w_q & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \end{cases}$

2.1.2 Relative Ranking Vector Calculations

For each criterion in the hierarchical structure, a Relative Ranking Vector (RRV) is computed to express the score of an alternative relative to the rest $N-1$. These vectors are, firstly, calculated for each KPI based on the corresponding RCMs. Assuming an RRV $rrv = [r_i]_{i=1}^{N}$, the r_i expresses the score of alternative i for a criterion. For a consistent result, given an RCM $\mathbf{A} = [a_{ij}]_{N \times N}$ for a KPI X, for the RRV of X, which is obtained by solving the problem, it should stand that $(r_i/r_j)_{N \times N} = \mathbf{A}$. Regarding the RRV computation of the attributes of the higher level, it is calculated based on the lower level RRVs and the weight values for each lower lever criterion.

Several methods have been proposed in the literature to estimate the relative ranking vector for the KPIs [72]. Given an RCM $\mathbf{A} = [a_{ij}]_{N\times N}$ the RRV can be obtained by the below methods that have been proposed:

- *Mean of normalized values*: This method calculates the elements of RRV $rrv = [r_i]_{i=1}^{N}$ as:

$$r_i = \frac{\sum_{j=1}^{N} a_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}} \tag{2.2}$$

- *Geometric Mean Method*: This method was introduced in [73] by Crawford and Williams, and it is widely used for the calculations of the AHP's priority vector. The elements of the relative ranking vector using this method, are calculated as follows:

$$r_i = \frac{\left(\prod_{j=1}^{N} a_{ij}\right)^{1/N}}{\sum_{i=1}^{N}\left(\prod_{j=1}^{N} a_{ij}\right)^{1/N}} \tag{2.3}$$

  As stated in [69], the vector which is computed by the geometric mean method, can equivalently be obtained as the argument minimizing the optimization problem:

$$\underset{r_1,\ldots,r_n}{\text{minimize}} \sum_{i=0}^{N}\sum_{j=0}^{N}(\ln a_{ij} + \ln r_j - \ln r_i)^2 \tag{2.4a}$$

s.t.:

$$\sum_{i=0}^{N} r_i = 1,\ r_i > 0, \forall i \tag{2.4b}$$

- *Eigenvector Method*: Saaty [74] proposed the eigenvector method to estimate the priority vector in the AHP. In specific, in [69] it stated that the priority vector should be the principal eigenvector of $\mathbf{A}$. Thus, the RRV $\mathbf{r}$ can be obtained as the solution of the following equation:

$$\begin{cases} \mathbf{A}\mathbf{r} = \lambda_{max}\mathbf{r} \\ \mathbf{r}^{\top}\mathbf{1} = 1 \end{cases} \tag{2.5}$$

  It is worth mentioning that λ_{max} is the maximum eigenvalue of $\mathbf{A}$, while $\mathbf{1} = (1,\ldots,1)^{\top}$.

2.2 Fundamentals of Graph Theory

In this section, a concise introduction regarding the basic terminology of graph theory aspects that will be used in the following chapters is given [75, 76].

2.2.1 Basic Graph Definitions

Definition 1. *A graph is a structure that is composed of vertices (or nodes) and edges (or links). It is denoted as a pair $G = (V, E)$ of sets, where $E \subseteq [V]^2$. Set V represents the vertices of G, while set E corresponds to the edges of the graph.*

A vertex is an individual element within a graph that represents a specific point or node. Vertices are often denoted using letters such as v_1, v_2, etc. An edge refers to the link or connection between two vertices in a graph, representing a relationship or association between them. Edges are commonly represented using pairs of vertices, such as (v_1, v_2).

Definition 2. *A directed graph, known, also, as a digraph, is a type of graph in which each edge has a specific direction associated with it. The direction of the connection is typically indicated using arrows or directed lines.*

The number of vertices that compose the set V is defined as the *order* of graph G and is denoted as $|G|$, while the corresponding number of edges in set E is denoted as $||G||$. To proceed with the rest definitions we note that the vertices x, y of G are *adjacent* or *neighbors* if there is an edge (x, y) in G. Also, the graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$.

Definition 3. *The degree $d_G(v)$ of a vertex v of graph G, is the number of edges in v, or equivalently the number of neighbors of v.*

Furthermore, assuming that $|V|$ is the number of vertices of G, we define the *average degree* of graph G as follows:

$$d(G) = \frac{1}{|V|} \sum_{v \in V} d_G(v) \tag{2.6}$$

It stands that the number of edges in G can be calculated as follows: $|E| = \frac{1}{2} d(G)|V|$.

2.2.2 Paths and Cycles

Given a graph $G = (V, E)$ the following definitions regarding paths and cycles are provided:

Definition 4. *A path P in a graph is a sequence of vertices (nodes) $\{v_1, v_2, \ldots, v_k\}$, where each pair of vertices v_i, v_{i+1} is connected by an edge (link). A path in which all vertices are distinct is called a simple path.*

In the case of a weighted graph, where a value is associated with every edge (link) of it, the *weight* or *cost* of a path is the sum of the weight values of the traversed edges, while the *length* of a path equals the number of edges in it. For a path $P = \{v_1, v_2, \ldots, v_k\}$, with $0 \leq i \leq j \leq k$, we write:

$$Pv_i := \{v_0, \ldots, v_i\},$$

$$v_i P := \{v_i, \ldots, v_k\},$$

$$v_i P v_j := \{v_i, \ldots, v_j\}.$$

Definition 5. *Given a path* $P = \{v_0 \dots v_{k-1}\}$ *and* $k \geq 3$*, we define as cycle the graph* $C := Pv_0$*, and it can be written as* $\{v_0, \dots, v_{k-1}, v_0\}$*.*

2.2.3 Connectivity

Definition 6. *We call a graph* G *connected if there is a path that connects every pair of its vertices.*

On the other hand, a graph is considered *disconnected* if there are at least two of its vertices that are not connected by any path. Furthermore, a *connected component* in G is a subgraph of G which is connected.
In the case of directed graphs, a graph is called *Strongly Connected* if a directed path connects any vertex to any another one. A graph is said to be *Weakly Connected* if replacing all the directed links with undirected links will lead to a connected graph.

Chapter 3

Multi-Objective Edge Cloud Selection for Network Service Deployment

3.1 General Setting

As mentioned earlier in the Introduction, innovative 5G verticals are composed of various time- and mission-critical applications that are developed as end-to-end services over heterogeneous distributed computing and network infrastructures. Towards network delay minimization and increased reliability and bandwidth savings, infrastructure providers offer computing, network, and storage resources at the edge of the network. Edge Computing, which has become the emerging service delivery paradigm [77], raises various service management and orchestration challenges. In this chapter, by the term *Edge Clouds (ECs)* we denote small-sized computing facilities directly connected to a wireless access point at the network edge. Under this setting, the dominant network architecture concepts of NFV and SDN continuously evolve to fulfill the various (non-)functional requirements for service orchestration of 5G verticals.

This chapter focuses on the deployment of 5G verticals in ECs. Specifically, deployments based on the concepts of network slicing and network services (NSes), which provide the desired isolation among tenants, are taken under consideration. The co-hosting of such service ecosystems creates opportunities for cross-service interactions within the same EC infrastructure, where an NS can consume an NS of a different tenant. The CSC enhances the overall performance in terms of throughput, latency, and potentially other KPIs, and creates opportunities for financial rewards through service leasing. In a similar manner, service components may be shared among different service operators, *i.e.,* a service orchestration aspect, known as inter-service operability. These emerging trends pave the way towards a next-generation NSM, where users are able to compose

custom service chains, while a service will be empowered to consume other co-located services in a controlled and secure manner [14]. The NSM should provide several automated functionalities to both service providers and consumers, while supporting the entire life cycle of the NSes, including service registration, discovery, instantiation, and termination. Furthermore, it must provide secure and automated mechanisms for the CSC establishment.

Every user can act either as a service provider or consumer. Thus, the NSM should provide essential mechanisms for service registration and discovery. The service registration should allow users to describe the features and requirements of their NSes in a consistent and open manner that can be used by the participating heterogeneous ECs. With this capacity, the ETSI NFV data model describes analytically the structure and parameters of the network services [78]. This model can be further enriched with application descriptors that provide QoS requirements. Upon service registration, the service discovery empowers users to identify the appropriate NSes for CSC. An NSM consists of geographically dispersed ECs for maximized footprint, with each one locally maintaining information about the CSC-available NSes. The service discovery is performed either centralized or in a distributed manner. A centralized approach commonly entails scalability limitations, while the efficiency of a distributed discovery mechanism is strongly affected by the underlying EC infrastructure.

Under this complex setting, the ECs of a single provider may have heterogeneous resources, which implies different features (*e.g.,* hardware acceleration, optimized CPU instruction sets, etc.), which, in turn, can lead to performance differentiation between ECs. Furthermore, the resources of ECs are limited, while resource allocation must be carried out such that service KPIs are met. Thus, prior to service component (*e.g.,* virtual network function) placement, the candidate ECs, assessed and proposed by the service discovery mechanism, should be evaluated based on multiple performance, cost, and support criteria [79]. Thereby, the final EC selection problem can be formulated as a multi-objective optimization problem, based on well-defined KPIs, which are common for every EC.

In this respect, we propose a distributed service discovery and multi-criteria EC selection mechanism for enabling 5G network slicing and CSC at the network edge. We assume that a service embedding request can be handed onto any EC, which is held responsible for the discovery and evaluation of candidate ECs, based on a multitude of criteria. The contribution of this work is twofold. First, a cache-based discovery protocol is designed, which aims at discovering the required services for CSC with fast convergence and controlled communication overhead. Second, we employ an MCDM approach to rank the candidate ECs based on the individual requirements of each service request. The distributed service discovery framework is evaluated using realistic EC-level topologies.

In comparison with both centralized and distributed service discovery solutions (*i.e.,* flooding), our proposed approach yields a significant reduction in the communication overhead, without any perceptible penalty in terms of discovery efficacy. Our evaluation results show that the proposed ranking mechanism is scalable with the number of ECs.

3.2 Related Work

Regarding the dominant aspects of network slicing, the interested reader can refer to [80]. This section presents the most relative studies regarding the establishment of the NSM, where service discovery techniques and provider or service selection mechanisms are adopted, for network service deployments in the context of 5G. Also, the most relative multi-criteria studies on cloud and network slice provider selection are presented.

Recently several projects have focused on the establishment of NSM, which provides service registration and discovery functionalities and enables cross-service communication. Aligned with ETSI NFV architecture [81], T-NOVA introduces the notion of the *Network Function Store*, which empowers clients to compose services out of individual network functions, supplied through this store [21]. Similarly, the FENDE Marketplace enables developers to on-board services as VNFs in a service catalogue, whereas users can select among the available VNFs and plug them into an instantiated slice on the underlying infrastructure [39]. Based on the concept of Resource Broker, the NECOS Marketplace facilitates resource discovery for instantiating network slices across multiple resource providers [27]. MESON advocates cross-slice communication between co-located slices in an EC, with respect to the requirements of both slice tenants [14]. Based on Open Source MANO (OSM) [82], the MESON orchestration provides CSC-enabled service discovery, EC selection, and slice placement. Considering the distributed service discovery across multiple providers or Points-of-Presence (within a single provider), the 5GEx hierarchical architecture of multi-domain orchestrators enables the exchange of high-level information with clients, service discovery and mapping, as well as VNF configuration and monitoring [83]. Also, in a trust-less multi-domain environment, Distributed Ledger Technology can facilitate multi-domain resource/service orchestration [84]. By implementing smart contracts, providers can offer, negotiate, and support the life cycle of their network assets based on specific trigger events.

Regarding the cloud provider evaluation and selection problem, Garg et al. [85, 86] defined a quality model for assessing Infrastructure as a Service (IaaS) providers that include various quantitative KPIs and attributes. The ranking system is based on AHP, which provides a weighted hierarchical structure of the attributes and KPIs and computes the relative ranking values of the candidate cloud providers. However, these approaches cannot process qualitative KPIs. Towards this direction, many fuzzy MCDM approaches have been used to solve various versions of the cloud selection problem. Papadakis et al. [87] applied the FAHP approach to select the appropriate provider in a cloud federation. This approach provided a Service Level Agreement (SLA) framework and processed evaluations of previous users of cloud infrastructure to produce a reputation score for each cloud provider. Based on SLA and monitoring data, a credibility mechanism was designed to alleviate the effect of malicious assessments. In [88], the authors proposed a modified fuzzy VIKOR approach for facilitating resource selection in a federation of heterogeneous testbeds. Since Fuzzy VIKOR is based only on fuzzy KPIs, any numeric KPI is transformed into a fuzzy one. Then, the evaluation of the users is compared with the assessment of a virtual user, which evaluates the best score for each KPI, in order to update the reputation value of the testbed. In this work, the user

of the infrastructure assigns their individual weights to the KPIs depending on the requirements of their service to be developed. CLOUDQUAL model[89] defined six quality dimensions for cloud services and evaluated them on three commercial storage clouds. For validation purposes, three criteria were used, i.e., correlation, consistency, and discriminative power, following IEEE standard for a software quality metrics methodology [90]. Li et al. [91] proposed a broker-based trust scheme for resource matchmaking across multiple clouds. This approach is based on various numeric service operators, such as security, availability, and reliability over specific time windows. Contrary to other studies, the trust attributes are not assigned subjectively by the users but they are computed by an entropy-based methodology. Finally, the authors of [79] proposed a multi-criteria method for placing network slices. Each infrastructure provider built reference slice blueprints and mapped them into many slice instances. Then, a vector of criteria is constructed for each candidate slice and the TOPSIS algorithm is used for the ranking of all candidate instances.

3.3 Distributed EC Selection Mechanism

This section presents the proposed Edge Cloud Selection mechanism, which is based on the multi-criteria decision-making method of FAHP. In a smart city context, cloud providers create a network of geographically dispersed and heterogeneous ECs. We consider an edge cloud infrastructure comprising geographically dispersed and heterogeneous ECs for the proposed edge cloud selection approach, as shown in Figure 3.1. Furthermore, the ECs are clustered in availability zones within specific areas, while each node in Figure 3.1 represents an EC. Each EC's resources are controlled by a Virtual Infrastructure Manager (VIM), which instantiates the services and assigns resources to them. A new network service embedding (NSE) request can be submitted to any EC, which selects the appropriate EC for hosting the NS. For the final selection, various functional, QoS, and cost requirements, as well as a set of desired CSC-enabled services are taken into account. Within each EC, several components are deployed to realize the service discovery and EC selection. In particular, the services of the NSM are stored in the *Service Catalogue* of each EC, whereas the *Service Cache* contains the available CSC services available in the adjacent ECs. The *Service Discovery Mechanism* (SDM) is responsible for sending query messages to its neighbors for CSC service discovery. Finally, based on an MCDM method, the ECRM evaluates the candidate ECs and selects the most appropriate one for NS deployment. The following sections describe in detail the functionality of the above components.

3.3.1 Service Catalogue and Service Cache

In this section, we focus on introducing and discussing two critical components of the proposed architecture that play a pivotal role in achieving the objectives of the distributed solution. These components are fundamental building blocks that provide the necessary information and functionality for the various mechanisms within the architecture to operate effectively. The Service Catalogue and Service Cache play integral roles in the NSM, serving as critical components that

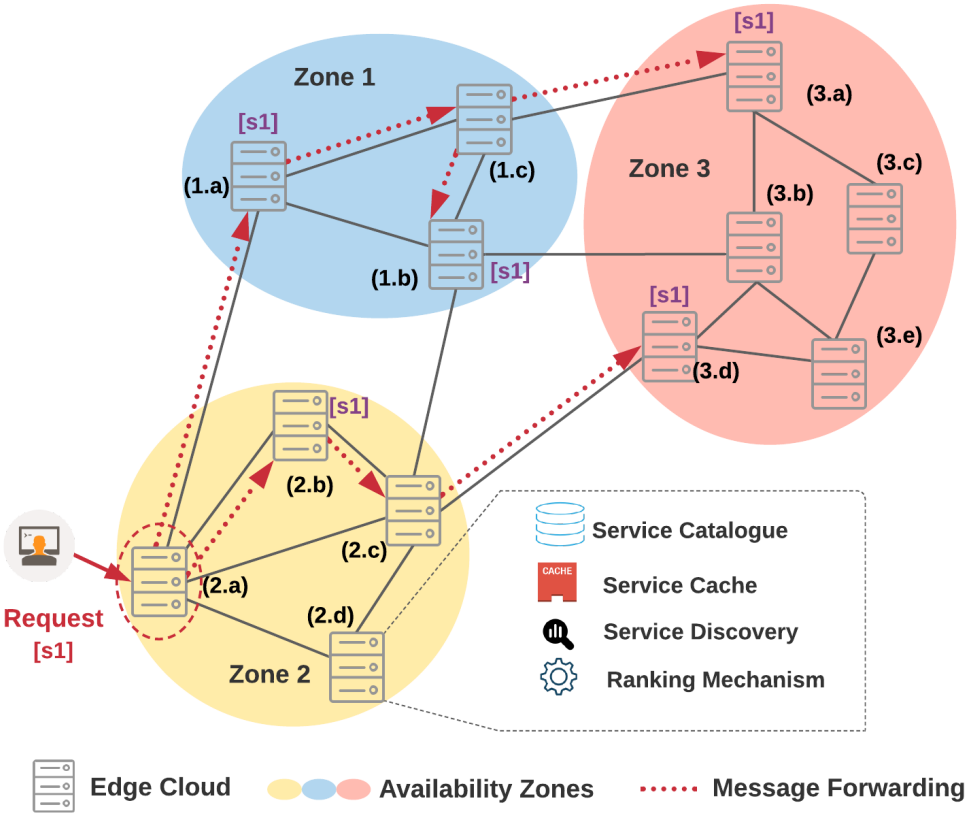


Figure 3.1: Distributed Service Discovery.

house essential information about the network services available for CSC. Each EC maintains its own Service Catalogue, which is a comprehensive list of CSC-enabled service instances deployed within its edge infrastructure. Following the MESON architecture, the service specification is defined using ETSI NFV network service descriptors and extended ETSI MEC Application descriptors [14]. These descriptors provide detailed information about the characteristics and capabilities of the services. The Service Catalogue serves as a central repository of available CSC service instances, allowing for efficient and streamlined service discovery. When a CSC request is received, the Service Catalogue is consulted to identify the relevant service instances that can fulfill the request. Additionally to the Service Catalogue, the EC's Service Cache holds the most recent information about the available CSC-enabled services of the neighboring ECs. The cache memory has a specific size and a FIFO-based replacement policy, in order to maintain minimal, but at the same time, enough, and, up-to-date information for efficient message routing, which implies an effective service discovery mechanism. An example of the cache entries for the ECs of

Fig. 3.1, is shown in Table. 3.1.

These components form the backbone of the NSM, enabling the discovery and utilization of network services through CSC. The Service Catalogue provides a comprehensive view of available services, while the Service Cache optimizes the matching process for faster and more efficient service selection. By leveraging standardized descriptors and well-defined specifications, the NSM ensures interoperability and compatibility between different services and ECs. Overall, the Service Catalogue and Service Cache are crucial components that enable the effective operation of the NSM, promoting collaboration and resource sharing among different network services in the edge computing environment. In the following, the proposed service discovery mechanism, which is based on the aforementioned components is described in detail.

3.3.2 Service Discovery Mechanism

The proposed architecture incorporates a Service Discovery Mechanism. When a user submits a request for a slice deployment in an EC, which is referred to as *Search Node*, the SDM is responsible for the query message forwarding. Instead of broadcasting routing query messages, the proposed SDM aims to forward a message to ECs that are most likely to host the requested CSC service instance. The message contains the requested CSC-enabled services and information about the forwarding policy, such as the known from the previous step ECs that have received the request and the Time-To-Live (TTL) value. The message routing is based on the cached information that each EC maintains about its neighbors' CSC-enabled services.

Starting from the Search Node, the discovery process follows the described policy, and a query message, which contains all the requested CSC-enabled VNFs is forwarded. Contrary to other service discovery techniques in the literature, like NECOS [27], in which the query message is transmitted to all the distributed infrastructures, the proposed method utilizes the local cache content of each EC to perform the message forwarding. The search depth in the EC network is determined by the TTL value. Algorithm 1 performs the discovery of the CSC-enabled services required by the user that submit the request, within the EC network. The algorithm's output is a search graph T, with root the Search Node and contains paths that lead to ECs that occurred as candidates for deployment infrastructures for the corresponding network service embedding request. At each step, the algorithm examines if the current TTL value is above the threshold. That is performed in lines 2-4, where the parameter $TTL_{current}$ has to be greater than zero. In this case, the algorithm forwards the query message based on the cache entries (line 5) and adds the corresponding edges in the search tree, while updating the current TTL value TTL' (lines 5-14).

As mentioned, an NSE request can be submitted to any EC, which is termed as *Search Node*, and the forwarding policy is based on the cached content of the *Search Node*. To better describe the forwarding procedure, which takes place in line 5 of Algorithm 1 of the query message, if one or more desired CSC-enabled services match with one or more cache entries, the query message is forwarded to these ECs. Otherwise, the *Search Node* forwards the message to all of its neighbors. In both cases, TTL is decremented by one. These forwarding rules are applied to any EC that

Table 3.1: EC Service Cache Example.

EC (2.a) Cache	EC (1.a) Cache	EC (1.c) Cache	EC (2.b) Cache	EC (2.c) Cache
(2.b) - $[s1, s4, s17]$	(1.c) - $[s16, s2, s1]$	(1.a) - $[s6, s14, s9]$	(2.a) - $[s5, s20, s7]$	(2.a) - $[s2, s15, -]$
(2.c) - $[s5, s8, s2]$	(1.b) - $[s3, s18, -]$	(1.b) - $[s3, s5, -]$	(2.c) - $[s2, s17, s5]$	(2.b) - $[s3, s11, s5]$
(2.d) - $[s2, s9, -]$	(2.a) - $[s4, s6, s12]$	(3.a) - $[s8, s12, s4]$	-	(2.d) - $[s7, s12, s6]$
(1.a) - $[s1, s6, s14]$	-	-	-	(3.d) - $[s3, s1, s8]$

Algorithm 1 CSC-enabled Service Discovery

```
1: procedure DISCOVERY(G, T, source, csc, TTL_current)
2:     if TTL_current < 0 then
3:         return T
4:     else
5:         forwardToNodes = Cache-based CSC Forwarding
6:         for node in forwardToNodes do
7:             if edge e(source, node) not in T then
8:                 ADD e(source, node) in T
9:                 TTL' = TTL_current − 1
10:                T=Discovery(G, T, node, csc, TTL')
11:            end if
12:        end for
13:        return T
14:    end if
15: end procedure
```

receives a query message, till TTL becomes zero. As mentioned, the already traversed ECs are encapsulated in the query message to avoid repetitive messages on the same nodes. Also, each EC that has received a query, responds to the sender EC with its most recent CSC-enabled services to update the respective cache entry. Finally, whenever at least one service match occurs, the EC sends a response to the *Search Node*, which contains information about the EC evaluation in the ECRM component.

Figure 3.1 illustrates an example of the SDM with three availability zones, namely *Zone 1, Zone 2, Zone 3*. Table 3.1 shows the cache entries of the ECs (2.a), (1.a), (1.c), (2.b) and (2.c). For this example, the TTL value is set to *3* and each cache entry contains up to three services for each neighbor EC. A NSE request is submitted to EC (2.a) (*Search Node*) that requires the CSC-enabled service `s1`. This service is hosted in the (1.a), (1.b), (2.b), (3.a) and (3.d) ECs. Based on its cache entries, the *Search Node* (2.a), forwards the request to the (1.a) and (2.b) ECs, as the `s1` matches with its corresponding entries for these ECs. Also, the (1.a) and (2.c) ECs forward the request based on the `s1` record in their caches. The ECs that are flooding the request are the (1.c) and (2.b), as their caches have no matching entry for `s1`. As shown in Table 3.1, the entry in the cache of (1.a) is out of date, as the `s1` is no longer hosted in the EC (1.c). When the ECs (1.b), (3.a) and (3.d) are reached, they do not forward further the request, as the TTL has zero value, so

their caches are not being included in Table 3.1. Every traversed EC sends two response messages. The first one is an update message to the neighbor EC, which has sent the request, and contains the three most recent CSC-enabled services to be cached. The second is a response message to the *Search Node* about the requested service(s). The ECs, hosting the requested service(s), are the candidate ECs and include to their responses to the *Search Node* the necessary information for the evaluation process.

3.3.3 Edge Cloud Ranking Mechanism

This section presents the proposed Edge Cloud Ranking Mechanism (ECRM). The ECRM is based on FAHP, which is a multi-criteria decision-making method. Taking into account the dynamic characteristics of an edge network, as well as the user's requirements for a network-slice deployment using CSC capabilities, the proposed mechanism is designed to maintain the necessary features, in order to meet these exact requirements and provide a ranking of the candidates, as they occurred after the service discovery procedure. Each EC maintains this mechanism, as it can possibly be the search node, where the initial user request is submitted. An NSE request can be submitted to any EC, which has a dedicated functionality for the edge cloud selection. In each EC, the ECRM selects the most appropriate EC for NS deployment and CSC establishment based on a set of functional, performance, and cost criteria and the set of desired CSC-enabled services. Extending the ranking mechanism of [24], the proposed ECRM is based on the AHP, which handles various types of numerical criteria. This provides the following enhancements: (i) the hard requirements are included in the structure of AHP in order to provide a more fine-grained evaluation regarding the hard requirements of the NSE request, (ii) the Virtual Edge Cloud (VEC) is defined to express the set of user's hard and soft requirements, in terms of ranking data. VEC profiles are defined based on the weight assignments and are used as a point of reference in the ranking process. In the following, for completeness, we provide a detailed description of the ECRM and its key components.

Ranking Criteria - Hierarchical Structure Definition:

According to the AHP, a hierarchical structure, which contains, suitable to the proposed architecture, KPIs and attributes, has to be defined. A KPI and an attribute differ in that a KPI refers to a specific technical or non-technical metric, while an attribute summarizes a group of KPIs of relevant metrics. Focusing on the automation of the deployment of a network slice and the establishment of cross-slice communication, the proposed architecture consists of two major attributes in the first level of the hierarchical structure. The Hard Constraints and the Soft Constraints. Contrary to a simple filtering solution, [24], the hard requirements (*i.e.,* the *Availability Zone* of the EC and the *CSC-enabled Services*) are included in the AHP model. This allows the ECRM to evaluate more EC candidates, which can be in different availability zones of the user's request. On the other hand, the *Soft Requirements* refer to performance, cost, and technical support criteria

Table 3.2: ECRM KPIs Definition [85]

KPI name	Definition
Availability	The percentage of time a customer can access the service.
Service Response	VM provisioning and booting time, assigning an IP address and starting application deployment.
Elasticity	The maximum number of compute units that can be provided at peak times.
Bandwidth	The minimum rate of data transfer, measured in Mbits per second.
VM Cost	Virtual machine acquisition cost.
Data Cost	Virtual machine ongoing cost for network resources.
Availability Zone	Multiple EC infrastructures within a distinct geographical area.
CSC-Enabled Services	A set of network services available to be shared among several tenants.

[24]. In the AHP, the ranking criteria are structured hierarchically and are separated in KPIs and attributes, in order to perform the selection between the alternative ECs. The key difference between a KPI and an attribute is that the first expresses a specific technical or non-technical metric, while the second summarizes relevant KPIs. In AHP structure, the KPIs are the leaf nodes of the hierarchy tree, while the rest are the attributes. Toward the automated network slice deployment, various relevant KPIs are defined, as those shown in Table 3.2, which are also widely used in many cloud provider selection studies [85, 89]. Figure 3.2 shows the hierarchical structure of our approach, which includes the most essential attributes and KPIs for the network slice case. The hierarchical structure is flexible and can easily be extended in terms of attributes and KPIs. The numeric performance and cost KPIs are defined by CSMIC [92], while the fuzzy KPIs of support are defined in [88].

AHP-based EC Ranking

Thereafter, the AHP phases towards the selection of the most suitable alternative EC for the NSE are exemplified in the following:

- **Phase 1 - Weight Assignment as User's Requirements:** As shown in Figure 3.2, the weights on the edges of the hierarchical structure correspond to the relative importance of each KPI and attribute in the ranking computation and represent the individual requirements of every service request. Towards the customization of the user's requirements, and based on the flexible customization that the AHP provides, the ECRM allows the user to assign the corresponding weight values for specific attributes and KPIs. In particular, the weights between the *Hard Requirements*, the *Soft Requirements*, the *Performance*, and their descendant are user-defined. Assuming that for a group of m siblings KPIs or attributes

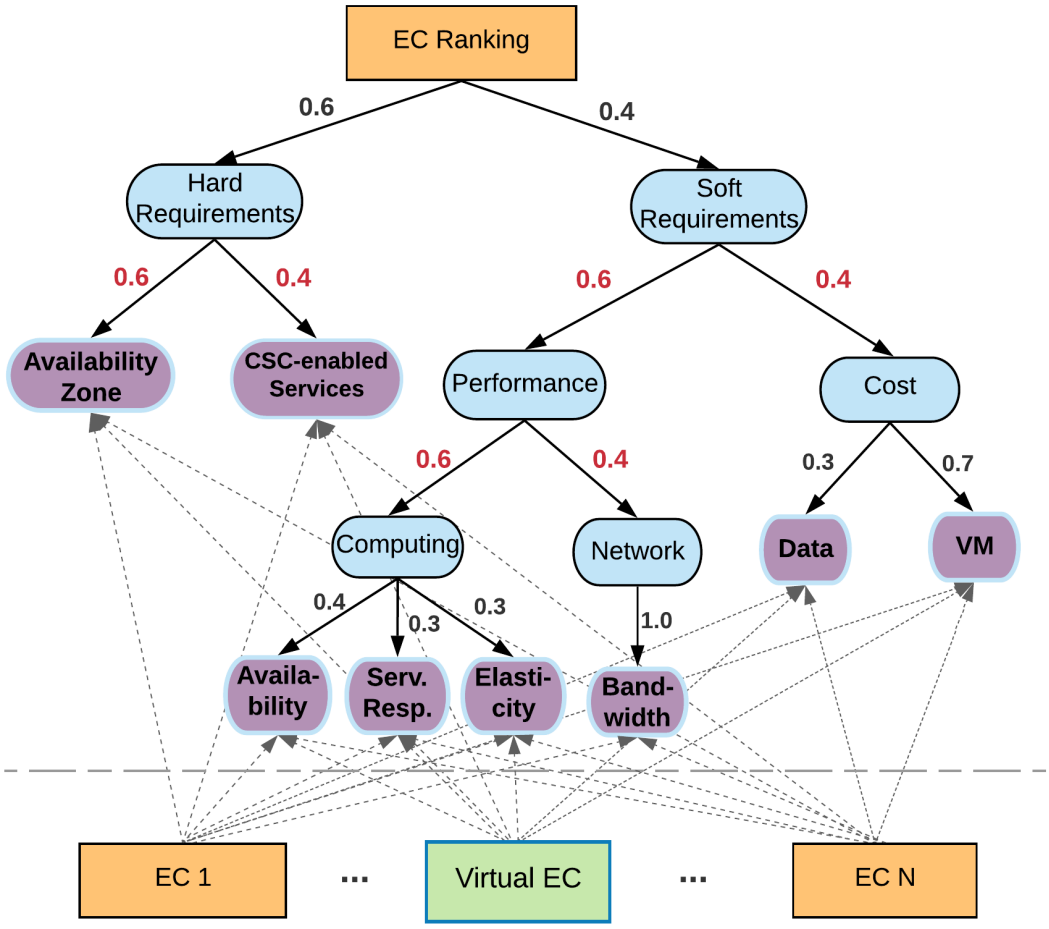


Figure 3.2: Hierarchical Model of Ranking Mechanism.

under an attribute A the assigned weight values are $w_j^{(A)}$, where $j = 1, \ldots, m$. Given the set $\mathbf{A}$ of attributes in the hierarchical structure, and $m^{(A)}$ the number of descendant KPIs or attributes of an attribute $A \in \mathbf{A}$, for a consistent weight assignment for each group of siblings attributes or KPIs, the sum of their weight values must be equal to one:

$$\sum_{j=1}^{m^{(A)}} w_j^{(A)} = 1, \forall A \in \mathbf{A}. \tag{3.1}$$

It is worth mentioning that $m^{(A)} = 0$ if A is a KPI.

Users are not allowed to define all the weight values in the structure, in order to avoid misconceptions in the ranking process. Thus, a Consistency Ratio (CR) is computed for

each group of sibling KPIs or attributes to avoid inconsistencies in the weight assignments [24]. The CR is the degree of randomness in the weight assignment between several sibling attributes. CR values less than 0.1 are acceptable to continue to the next phase, otherwise, the experimenter must correct the assigned weights [93].

- **Phase 2 - Relative Comparison Matrix Computation:** After the discovery process, each candidate EC, in its response message to the *Search Node*, advertises its data for the corresponding KPIs of the hierarchical structure, which are taken into account in the ranking process. Given the hierarchical structure, the KPI values for each candidate EC, and the weight assignments (that express the user's requirements) of the corresponding NSE request the scope of the AHP process is to generate a relative ranking vector (RRV) for each attribute of the hierarchy, from which the top level attribute, the *EC Ranking*, corresponds to the overall ranking of the ECs that took place in the ranking process. The AHP is a bottom-up process, and the first element that is mandatory to compute the RRV for each KPI, based on the corresponding advertised values is the Relative Comparison Matrix (RCM). Each element of this matrix expresses the pairwise comparison between two candidate ECs for a specific KPI.

  Assuming that n ECs take place in the ranking procedure and a_i, where $i = 1, 2, ..., n$ is the value that is advertised from the EC i for the KPI A. Then, the RCM matrix $\mathcal{X}$ of the KPI A is computed as follows,

$$\mathcal{X}_A = \begin{bmatrix} 1 & a_1/a_2 & \dots & a_1/a_n \\ a_2/a_1 & 1 & \dots & a_2/a_n \\ \vdots & \vdots & & \vdots \\ a_n/a_1 & a_n/a_2 & \dots & 1 \end{bmatrix} \tag{3.2}$$

  There are KPIs of different types, so the necessary calculations are used for each different case. These are presented in Chapter 2 *Background*, Table 2.1.

- **Phase 3 - Relative Ranking Vector Computation:** At any level of the hierarchy, for each attribute, an RRV is computed, which provides a normalized score for every candidate EC for the specific attribute. Starting from the bottom level of the hierarchy, the upper-level vectors are computed. The top-level attribute corresponds to the overall ranking of the ECs. Eventually, the highest-scored EC is selected for the service deployment.

  Especially, for the relative ranking vectors of the KPIs of the hierarchy, the corresponding RCMs are utilized. As mentioned in Chapter 2, there are several methodologies to obtain the RRVs of the KPIs. For the proposed ranking mechanism, the *Means of normalized values* methodology is adopted. Thus, for a KPI A with RCM $\mathcal{X}_A = [x_{ij}^{(A)}]_{n \times n}$ the RRV $\mathbf{R}_A$ is computed as follows:

$$\mathbf{R}_A = [r_1^{(A)}, r_2^{(A)}, \ldots, r_n^{(A)}]^\top, \text{where } r_i^{(A)} = \frac{\sum_{j=1}^{N} x_{ij}^{(A)}}{\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}^{(A)}}. \tag{3.3}$$

For an RRV $\mathbf{R}_A$ it stands that: $\sum_{i=1}^{n} r_i^{(A)} = 1$. Following the bottom-up process of the AHP, regarding the RRVs of the upper layers, its computation is based on the RRVs of the lower levels of the hierarchy and the weight values for each group of sibling KPIs and attributes. Considering a parent attribute of the hierarchy, denoted as A_{par} with m sibling sub-attributes (or KPIs) $S_1, S_2, \ldots S_m$ with corresponding RRVs $\mathbf{R}_{S_1}, \mathbf{R}_{S_2}, \ldots, \mathbf{R}_{S_m}$, and $w_{s_1}, w_{s_2}, \ldots, w_{s_m}$ the weight values of the edges that connect the A_{par} with its descendant, the RRV, in this case, is computed as:

$$\mathbf{R}_{A_{par}} = \begin{bmatrix} r_1^{s_1} & \ldots & r_1^{s_m} \\ r_2^{s_1} & \ldots & r_2^{s_m} \\ \vdots & & \vdots \\ r_n^{s_1} & \ldots & r_n^{s_m} \end{bmatrix} \begin{bmatrix} w_{s_1} \\ \vdots \\ w_{s_m} \end{bmatrix} = \begin{bmatrix} r_1^{(A_{par})} \\ \vdots \\ r_n^{(A_{par})} \end{bmatrix} \tag{3.4}$$

This bottom-up procedure leads to the top level RRV, which contains the final ranking result for the ECs.

VEC Candidate

The above process evaluates the candidate ECs based on the user's requirements and the data that is advertised from the ECs for the corresponding KPIs. So, the ranking occurs from the values of the relative ranking vector as it resulted from the AHP calculations. However, the result that is returned to the user should be provided as a clear suggestion as to which ECs are considered suitable for the network slice deployment. Thus, it is important to define a point of reference in the final ranking result, as some of the candidate ECs might be better off being rejected from the ranking mechanism. Hence, in the proposed ranking mechanism, we introduce the concept of a VEC. The VEC can be a part of the ranking procedure, in order to be used as a point of reference. The addition of the VEC must be done in such a way as to ensure the efficiency of the ECRM, the clarity of the ranking result, and the automation of the whole process. To achieve that, two major requirements have to be addressed. First, in order to include the VEC as an additional candidate in the ranking process the appropriate values in the respective KPIs must be defined. Secondly, the different VEC profiles have to be determined according to the user's requirements, that submits the corresponding NSE request. As we mentioned above, the user is able to assign weights to specific KPIs and attributes, aiming to express the requirements for the NSE request. According to those criteria importance assignments, a perform an importance-based clustering, which aims to identify clusters of weight values that determine specific requirements, which can best express a general set of users, thoroughly. Afterwards, for each cluster of weights, a VEC profile is defined. Each VEC

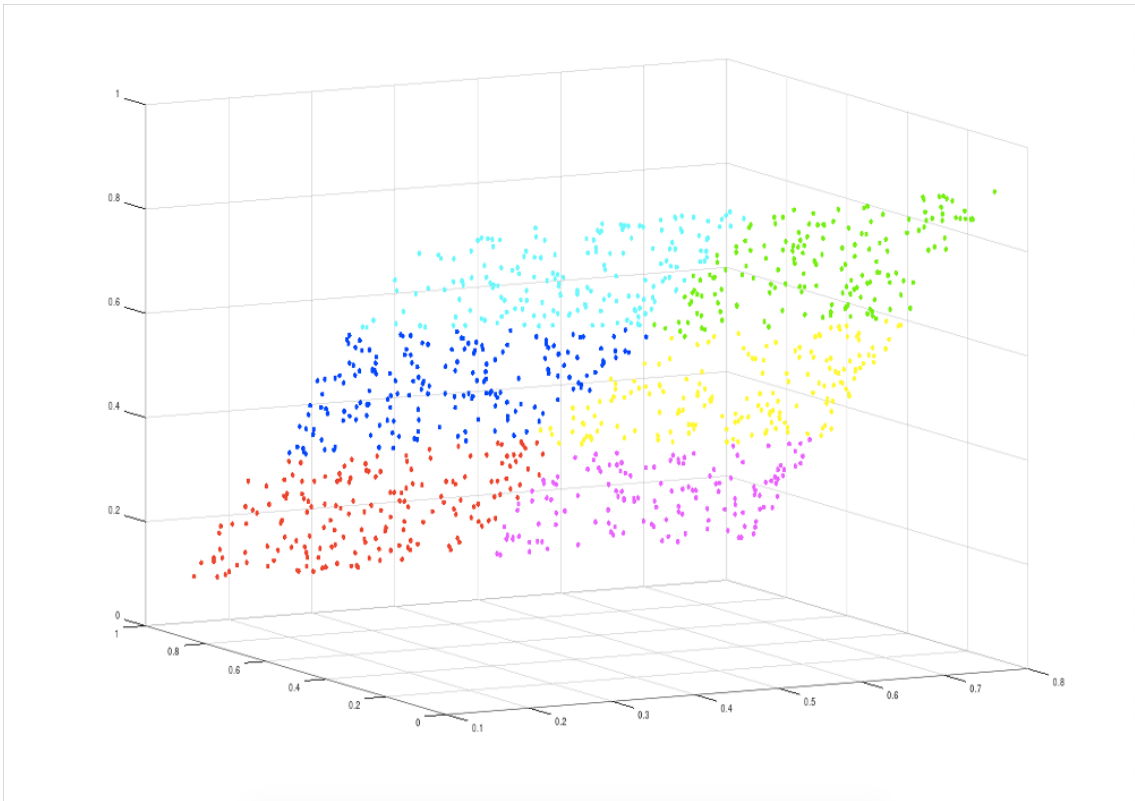


Figure 3.3: k-Means weight-based Clustering for VEC profiles.

profile corresponds to a specific set of KPI values. Using the k-Means algorithm [94], which is a well-known unsupervised clustering algorithm, aiming at partitioning n observations into k clusters in which each observation belongs to the cluster with the nearest centroid point. The clustering parameters that are used in the proposed VEC clustering are the weight values of the attributes *Cost* (relative to the *Performance*), *Computing Performance*, and *Network Performance*. A data set of thousands of different weight assignments is populated for those attributes, which is then provided in the k-means clustering algorithm to produce the centroids $(\tilde{w}_{cost}, \tilde{w}_{compp}, \tilde{w}_{netp})$. In that way, six different clusters of VEC occurred: *Low Cost – High Computing Performance*, *Low Cost – High Network Performance*, *Low Cost – General Performance*, *Indifferent Cost – General Performance*, *Indifferent Cost – High Computing Performance* and *Indifferent Cost – High Network Performance*. Thus, after the weight assignment from the user, a specific profile of VEC is chosen to take place in the ranking process. Each of the VEC clusters has predefined values for the KPIs, in a way suitable to each category. Especially for the Hard Constraints KPIs, the values are the same as those that the user requires. Thus, after the weight assignment from the user, a specific profile of VEC is chosen to take place in the ranking process. Each of the VEC clusters has predefined values for the KPIs, in a way suitable to each category. Especially for the Hard

Table 3.3: Centroids of the VEC Profile Clusters Based on the Weight Assignment

VEC Profile Cluster	Centroid
Low Cost – High Computing Performance	$(0.64016, 0.78311, 0.21689)$
Low Cost – High Network Performance	$(0.64194, 0.24705, 0.75295)$
Low Cost – General Performance	$(0.65088, 0.54184, 0.45816)$
Indifferent Cost – General Performance	$(0.26919, 0.47041, 0.52959)$
Indifferent Cost – High Computing Performance	$(0.28643, 0.76259, 0.23741)$
Indifferent Cost – High Network Performance	$(0.33521, 0.22478, 0.77522)$

Constraints KPIs, the values are the same as those that the user requires. The VEC is evaluated as a candidate EC and its relative score implies that ECs with higher ranking satisfy the user requirements.

3.4 Evaluation

In this section, we evaluate the efficiency of the proposed edge cloud selection mechanism. To this end, we consider a network of 300 ECs, subdivided into three availability zones (*Zone1*, *Zone2*, and *Zone3*). Based on real network topologies of regional providers [95], the size of these availability zones is set to 50, 100 and 150 ECs respectively. A pool of 20 services for enabling CSC is available at every EC. Each EC can host from 5 to 15 CSC-enabled services. Furthermore, each EC advertises its values on the KPIs of the AHP model. These values are randomly selected using a uniform distribution. In order to highlight the impact of the important system parameters on the results of discovery and ranking, TTL varies from 1 to 8, while the size of Service Cache ranges from 3 to 5. In the following experiments, for each TTL value, 90 NSE requests with random values of hard and soft requirements are generated and uniformly assigned to the ECs of the three availability zones. Each of those requests is being handled by the proposed mechanism for the different cache sizes. Furthermore, one or two CSC-enabled services are required in every NSE request.

The first experiment aims at assessing the SDM efficiency. In particular, the SDM is compared against a standard discovery technique, termed as *Flooding*, which broadcasts the query message to every neighbor of the sender EC. *Flooding* relies on the Breadth-First-Search (BFS) approach [96], which prevents repeating queries on an EC. In the following, we compared these two techniques in terms of generated communication overhead and network utilization. Also, we compared our solution with the NECOS centralized approach for discovery in cloud resource marketplace [27], in terms of the total number of exchanged messages. The NECOS centralized broker is responsible for sending the NSE request in every EC and processing the corresponding responses.

Regarding the ECRM evaluation, we demonstrate a ranking of three ECs in order to quantify the potential gains from the use of the Hard Requirements in the AHP (instead of employing a simple filtering solution). We further assess the influence of the VEC in the ranking results.

Finally, we provide additional remarks about the overall performance of the proposed mechanism.

3.4.1 SDM Evaluation

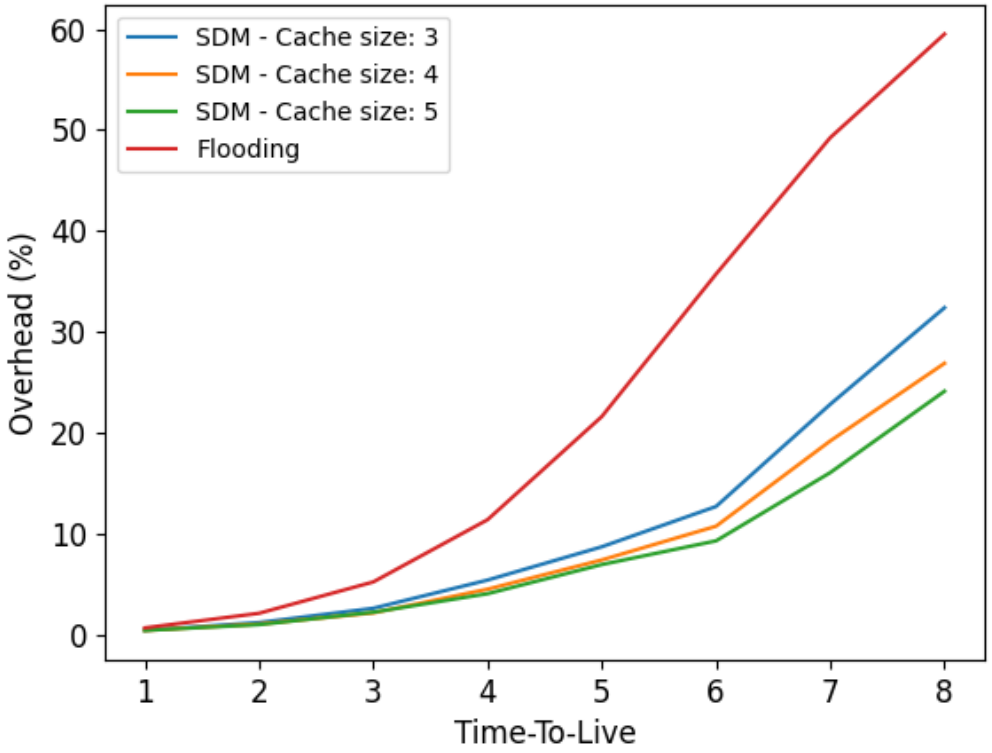


Figure 3.4: Generated Communication Overhead.

In this experiment, we compared the efficiency of the SDM against the *Flooding* technique with diverse values of the TTL and the Service Cache size. In this respect, we measure the communication overhead as the percentage of utilized links. As shown in Figure 3.4, the proposed SDM severely reduces the utilized links in the search-tree, compared to *Flooding*. More precisely, SDM yields a communication overhead of 9.5% on average, whereas the respective value for *Flooding* is 23.2%. This low communication overhead is also reflected by the number of messages exchanged. In particular, the SDM generates 138 messages, as opposed to *Flooding* at which 220 messages are exchanged (*i.e.,* 38% reduction for SDM). We note that, in the SDM, each EC generates two messages, the cache update message and the response to the *Search Node*, while the *Flooding* approach generates only one response message. Regarding the comparison with NECOS, we conducted experiments by setting the TTL equal to 8 to measure the generated messages in a search depth equal to the centralized approach. For different cache sizes (3 to 8), 30 requests are

submitted from each of the availability zones. The centralized broker approach overall requires 600 request/response messages, while SDM generates an average of 304 messages, which further corroborates the efficiency of SDM compared with centralized solutions in terms of communication overhead.

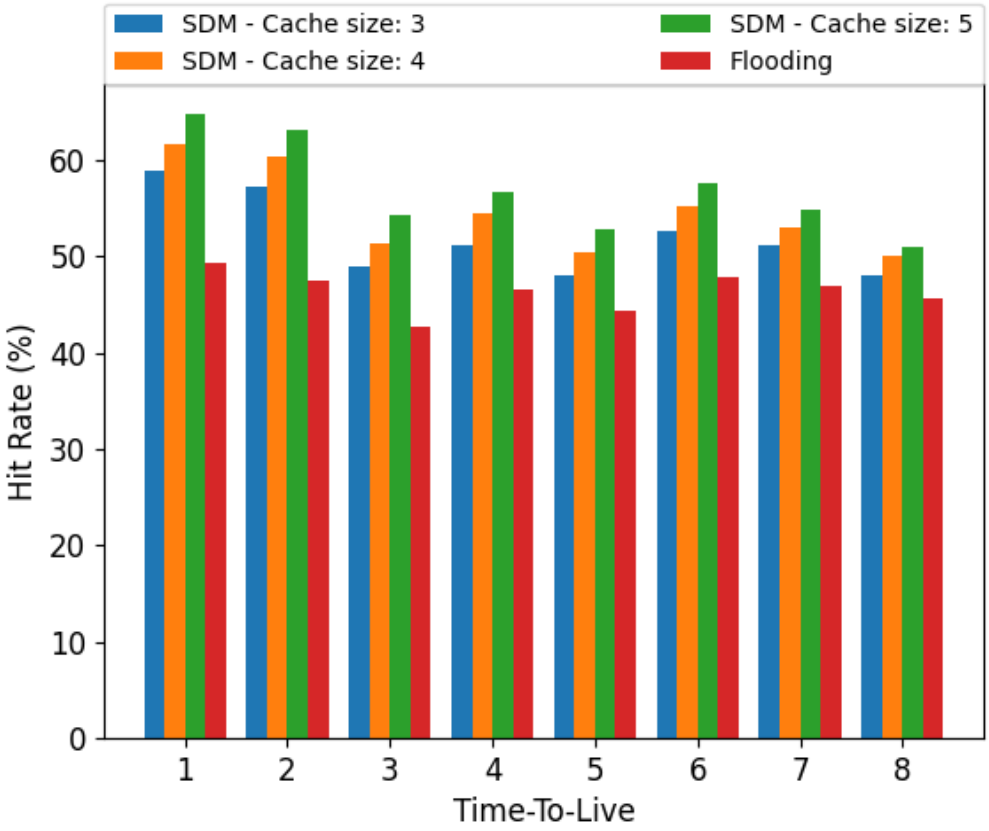


Figure 3.5: Cache Hit Rate.

Figure 3.5 illustrates the discovery capability of SDM (with a range of TTL values and cache sizes) and *Flooding*. We use the *Cache Hit Rate* as metric, which corresponds to the ratio of the discovered ECs with at least one required CSC-enabled service over the total number of queried ECs. The average *Cache Hit Rate* for the SDM and *Flooding*, across all conducted tests, are 56% and 47%, respectively. Furthermore, 76% of the candidate ECs evaluated by the ECRM are in the availability zone of the *Search Node*. Due to more queried ECs, this percentage is lower for the *Flooding* approach. Regarding the cache size, the larger the cache is the hit rate is slightly better $(2-3\%)$. From our tests, we infer that an efficient cache size is approximately 20% of the total available CSC-enabled services.

Table 3.4: Advertised EC KPI Data - VEC Data

KPIs	Metric	EC1	EC2	EC3	VEC
Availability Zone (0.6)	boolean	1	1	0	1
CSC-enabled Ser. (0.4)	#	2	1	2	1
Serv. Availability (0.4)	%	95	85	85	90
Service Response (0.3)	ms	15	20	15	10
Elasticity (0.3)	# VMs	3	1	2	2
Bandwidth (1.0)	Mbps	20	30	30	20
Data Cost (0.3)	$	0.3	0.25	0.25	0.5
VM Cost (0.7)	$	5	15	10	20

3.4.2 ECRM Evaluation

The first two columns of Table 3.4 include the identifier, the weight, and the type of KPIs of the AHP model. The following three columns contain the values of these KPIs for three ECs, whereas the last column contains the KPI values of the VEC, according to the chosen profile and the user's hard requirements. Following a bottom-up approach, a ranking vector is computed for the ECs, including the VEC. The weight values are, also, shown in Figure 3.2 at the edges of each attribute. The blue-colored rows are the KPIs of the Computing Performance attribute, the Bandwidth is the KPI of the Network Performance attribute, and the red-colored KPIs refer to the Cost Attribute. The user's requirements for the KPIs about the Hard Requirements are also shown in Table 3.4. About the Performance and Cost attributes the weights are: (0.6) for the Performance, (0.4) for the Cost, (0.6) for the Computing Performance and (0.4) for the Network Performance. As a point of reference, the VEC profile of *Indifferent Cost – High Computing Performance* is used in the ranking process. In order to provide a numerical example of the AHP calculation, next we present the calculation of the RRV for the KPI *Service Availability* and the attribute *Cost*. At first, we calculate the RCM of the *Data Cost*:

$$\mathcal{X}_{avail} = \begin{bmatrix} 1 & \frac{95}{85} & \frac{95}{85} & \frac{95}{90} \\ \frac{85}{90} & 1 & \frac{85}{85} & \frac{85}{90} \\ \frac{85}{90} & \frac{85}{85} & 1 & \frac{85}{90} \\ \frac{90}{95} & \frac{90}{85} & \frac{90}{85} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1.117 & 1.117 & 1.055 \\ 0.894 & 1 & 1 & 0.944 \\ 0.894 & 1 & 1 & 0.944 \\ 0.947 & 1.058 & 1.058 & 1 \end{bmatrix}$$

. Then, based on equation (3.3), the relative ranking vector $\mathbf{R}_{s\_avail}$ of the KPI is computed:

$$\mathbf{R}_{avail} = \left[\frac{4.289}{16.028}, \frac{3.838}{16.028}, \frac{3.838}{16.028}, \frac{4.063}{16.028}\right]^{\top} \Rightarrow$$
$$\mathbf{R}_{avail} = [0.27,\ 0.24,\ 0.24,\ 0.25]^{\top}$$

Based on the same method the RRVs of the KPIs *Service Response* and *Elasticity* are calculated:

$$\mathcal{X}_{resp} = \begin{bmatrix} 1 & 0.75 & 1 & 1.5 \\ 1.33 & 1 & 1.33 & 2 \\ 1 & 0.75 & 1 & 1.5 \\ 0.67 & 0.5 & 0.67 & 1 \end{bmatrix} \xRightarrow{(3.3)} \mathbf{R}_{resp} = [0.25,\ 0.33,\ 0.25,\ 0.17]^\top$$

$$\mathcal{X}_{elast} = \begin{bmatrix} 1 & 3 & 1.33 & 1.33 \\ 0.33 & 1 & 0.5 & 0.5 \\ 0.67 & 2 & 1 & 1 \\ 0.67 & 2 & 1 & 1 \end{bmatrix} \xRightarrow{(3.3)} \mathbf{R}_{elast} = [0.363,\ 0.127,\ 0.255,\ 0.255]^\top$$

. After the computation of the KPIs' RRVs, the latter is computed for the attribute of the upper level of the hierarchy, which summarizes the corresponding KPIs. In the particular example, this attribute is the *Computing Performance*. The RRV of this attribute is computed as follows:

- First the weight values of the involved KPIs define the importance vector:

$$\mathbf{w} = [w_{avail}, w_{resp}, w_{elast}] = [0.4,\ 0.3,\ 0.3]^\top$$

  .

- Then, the relative ranking vector for the attribute A is produced by matrix multiplication, in which the RRVs of the KPIs construct a $n \times m^{(A)}$, where $m^{(A)} = 3$ in this example:

$$\mathbf{R}_{compPerf} = \begin{bmatrix} \mathbf{R}_{avail} & \mathbf{R}_{resp} & \mathbf{R}_{elast} \end{bmatrix} \mathbf{w} \Rightarrow$$

$$\mathbf{R}_{compPerf} = \begin{bmatrix} 0.27 & 0.25 & 0.363 \\ 0.24 & 0.33 & 0.127 \\ 0.24 & 0.25 & 0.255 \\ 0.25 & 0.17 & 0.255 \end{bmatrix} \begin{bmatrix} 0.4 \\ 0.3 \\ 0.3 \end{bmatrix} = [0.292,\ 0.233,\ 0.247,\ 0.228]^\top$$

  .

Following the AHP method, the overall ranking result is: $[0.322, 0.244, 0.196, 0.236]$, which means that the EC ranking from the best to the worst is: (1) EC1, (2) EC2, (3) VEC, (4) EC3. As such, the EC3 is considered as an unacceptable solution, since it is ranked below the VEC. The inclusion of the hard requirements in the AHP model and the addition of the VEC profile in the evaluation lead to a more fine-grained ranking, promoting the candidate ECs that better satisfy the user's requirements.

Another important remark on the overall performance of the proposed mechanism is derived. For all experiments, evaluating the candidate ECs discovered by the *Flooding* technique provides at least one solution with a higher ranking than the VEC. For the candidate ECs discovered by the SDM, the ECRM provides at least one solution that satisfies the user's requirements for

99.5% of the experiments. This result implies that most of the selected ECs for NS deployment are within the availability zone of the *Search Node* and that the proposed mechanism achieves identical efficiency with the greedy *Flooding* approach with significantly lower communication overhead.

3.5 Chapter Summary

Enabling Network Service Marketplaces is crucial for reaping the benefits of 5G network technologies. To this end, efficient CSC at the network edge can pave the way for service provisioning and cross-service interactions. In this work, we presented a distributed EC selection framework towards 5G Network Service Marketplaces. The proposed framework encompasses (i) a service discovery mechanism, which yields low communication overhead and fast convergence, stemming from caching and the efficient reduction of search space, and (ii) a multi-criteria ranking mechanism, which evaluates candidate ECs in terms of service deployment.

The determination of sophisticated VEC profiles through Machine Learning techniques could improve the representation of the individual requirements of the NSE requests. Potential extensions of the framework include the NS embedding across multiple domains in a distributed manner.

Chapter 4

Time-Efficient Distributed Virtual Network Embedding For Round-Trip Delay Minimization

4.1 General Setting

As discussed in the introductory Chapter 1 of the dissertation, modern 5G applications rely on various IoT devices, often generating vast amounts of data. Usually, these IoT devices have limited computational resources and energy capacity, thus, they are not able to locally perform complex processing. On the other hand, the Edge Computing paradigm promises to mitigate the above restriction by providing computing infrastructure in the proximity of the end-users. Furthermore, processing at the network edge results in reduced network latency compared with processing on traditional cloud resources. However, contrary to the coarse cloud case, the edge resource management must be fine-grained due to the limited edge resources and the strict time constraints of IoT applications [77]. In this context, many resource orchestration platforms have been developed that rely on either VMs or containers, such as OpenStack [44] and Kubernetes [45], and they provide several functionalities about service embedding, resource allocation, and networking.

NFV architecture enabled the execution of network services as VMs on common hardware and therefore replaced any proprietary middle-box. An application consists of individual services, namely VNFs. An SFC includes several VNFs with predefined execution orders and specific computing and network requirements. The typical VNE problem focuses on embedding the SFC on the substrate network. Under this setting, the network traffic is directed from an ingress VNF to the egress one, while the response traverses the same path in the opposite direction. ETSI NFV [19] and ETSI MEC) [10] are the two dominant reference architectures that are used for the realization of the VNE in cloud and edge infrastructure.

However, with the advent of 5G and IoT technologies, the formulation of the VNE requires two key modifications. At first, the early definition of VNF strictly referred to network services (routing, firewall, and load balancing etc.). Nowadays, the VNF definition should be more application-oriented and include any type of service. Secondly, for IoT applications, which usually include online machine learning processing [47], the forward traffic consists of raw data, while the backward traffic includes results and models that must be sent back to the end devices that generate the network traffic. This possibly requires different forward and backward paths for the SFC.

Edge Computing provides geographically spread small-scale resources, defined as Edge Clouds, that create a distributed cloud environment with multiple stakeholders [60]. As described earlier, this enables the creation of Network Service Marketplaces, which allows subscribers to act both as service providers and consumers [28]. In this complex scenario, an already deployed VNF can be shared among different tenants in order to reduce the allocated resources in an EC and optimize the exchange of traffic between them. Under this setting, a SFC includes pre-installed VNFs that must be taken into consideration in the VNE solution.

Focusing on IoT applications deployed on distributed edge infrastructure, in this Chapter a distributed virtual network embedding mechanism that takes into account the above characteristics of the SFC and the distributed multi-stakeholder edge environment is proposed. More specifically the scientific key contributions regarding this problem are the following,

- A heuristic that undertakes the optimal initial mapping of parts (VNFs) of a Virtual Network within an EC infrastructure considering the pre-deployed shared-VNFs, focusing on minimizing the infrastructure's resource utilization.

- Assuming that the forward and the backward paths of a VN are different, the proposed shortest-path-based algorithm finds the Distributed Virtual Network Embedding solution that minimizes the round-trip delay.

- The numerical results illustrate that the proposed distributed virtual network embedding algorithm computes the optimal or near-optimal solution, especially combined with the initial VN mapping heuristic, in a limited time compared to relevant existing state-of-the-art studies.

4.2 Related Work

This section presents a comprehensive overview of the most relative studies in the literature. For presentation purposes these studies are classified in three main categories, (i) shared VNF-based VNE approaches, (ii) distributed VNE approaches, and (iii) shortest path-based VNE approaches.

Shared-VNFs host common services, that can be consumed simultaneously by different tenants. A primary requirement of shared-VNF is the isolation between different SFCs that share such VNF. Aligned with ETSI NFV and ETSI MEC reference architectures, MESON platform provides a secure mechanism for communication between slices of different tenants in order to save computing resources within an EC and reduce outgoing network traffic [14]. Papadakis et al. [15]

proposed a blockchain-based mechanism for cross-service communication that provides registration, searching, leasing, and billing functionalities over multiple ECs in the context of Network Service Marketplaces. Similarly, based on distributed ledger technology, the FENDE marketplace relies on shared-VNFs and facilitates the subscribers to deploy tailor-made SFCs over multi-cloud infrastructure [39].

Edge Computing enables the deployment of network services in small-scale infrastructure close to the end-users. Contrary to the cloud case, this infrastructure is geographically spread, and therefore, the VNE problem has been transformed to DVNE one. For MEC/Cloud environment, Zheng et al. [97] proposed a hybrid form of SFC with dissimilar forward and backward paths. This study focused on latency minimization and it proposed a brute force method on an augmented graph to select the shortest path. Considering a distributed cloud environment with three types of stakeholders (i.e., subscribers, service provides, and infrastructure providers), the authors in [98] formulated the DVNE problem as an Integer Linear Programming to both maximize the number of accepted SFC requests and satisfy the subscribers' requirements. Towards this direction, a pre-processing stage aims at rejecting those requests that are infeasible to be accomplished by the system, defining incompatibilities between VNFs and data centers, and defining subscriber preferences that are used in the objective function of the optimization model. Pei et al. [99] formulated the DVNE problem as a Binary integer Programming model. Subsequently, they proposed two algorithms for minimizing the embedding costs. Initially, the VNF instances are placed by a shortest-path algorithm in a multi-layer graph. Then, the placed VNFs are released using their utilization rate and a threshold according to workload variation. Constraints that refer to delay, reliability, mobility, and battery requirements, of mobile end-devices which act as an extension of the cloud and edge computing infrastructure are taken under consideration for the VNF placement by the authors in [100]. The problem is formulated as a cost-minimizing VNF placement optimization. A heuristic based on the fractional optimal solution of a bin packing variant is proposed to obtain the near-optimal VNF placement solutions while ensuring scalability in terms of reducing the convergence time, as occurred from simulations under a mobile robots scenario.

For either VNE or DVNE problem, many studies proposed shortest path-based models that allow computing a near-optimal solution with low complexity. The authors in [101] focused on the trust-aware VNE problem. For each VNF and network edge, they assumed trustworthiness requirements and formulated a path-based model to integrate network policies. In order to enhance the scalability of the model, they included the k-shortest paths of an augmented graph to find the optimal solution. Authors in [102], proposed a constrained shortest path algorithm to deal with the exponential nature of the VNE problem. They introduced the *Neighborhoods Method*, which utilizes dynamic programming and branch-and-bound exhaustive search, while search space reduction techniques are, also, considered. This method achieves a theoretical reduction of computational complexity compared to alternative exhaustive search solutions. Evaluation results certify the scalability and flexibility of the proposed algorithm when compared to other path-finding-based algorithms. The VNE problem in SDN is discussed in [103]. The authors provided a multi-objective

Table 4.1: Comparison of related studies.

Categorization	Related Studies										Proposed Study
	[14]	[15]	[104]	[97]	[98]	[99]	[100]	[101]	[102]	[103]	
Shared VNF-based VNE	✓	✓	✓	x	x	x	x	✓	x	x	✓
Distributed VNE	x	x	x	✓	✓	✓	✓	x	✓	✓	✓
Intra-EC VNE	✓	✓	✓	x	✓	x	x	x	x	x	✓
Path-based VNE	x	x	x	✓	✓	✓	✓	✓	✓	x	✓
Round-Trip Delay	x	x	x	✓	x	x	x	x	x	x	✓
Compute Resources	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Network Bandwidth	x	x	✓	✓	✓	✓	✓	✓	✓	✓	✓
Order Constraint	x	x	x	✓	✓	✓	✓	✓	✓	x	✓

optimization formulation of the VNE problem, towards, firstly, the minimization of the network load and secondly, the maximization of the embedding reliability, under the constraints of virtual network requirements and the resource characteristics of the substrate network. The problem is separated into two corresponding sub-problems. For the sub-problems solution, a two-stage VNE algorithm is proposed using slight modifications for each case. The authors assigned performance metrics to both virtual nodes and substrate nodes to perform the mapping greedily, while the virtual links are embedded by applying Dijkstra's algorithm in the weighted graph representation of the substrate network, to determine the shortest path between the source and the destination substrate node of one virtual link. Then, examining the distance between the feasible solutions and the locally optimal solutions, they formulated a single-objective optimization problem and solved it to obtain the global VNE strategy.

Table 4.1 summarizes the achieved goals of the aforementioned studies and highlights the differences with the proposed solution. Contrary to our work, the shared VNF-based studies [14, 15] focused mainly on the service discovery and the establishment of cross-slice communication in a single EC, while the intra-EC VNE is performed by default component of the VM orchestrator (i.e., OpenStack). On the other hand, FENDE marketplace [104] focused on the synthesis of the customized SFC and the VNE solution is handled by a local Virtualized Infrastructure Manager. Regarding the distributed VNE approaches, Cappanera et al. [98] proposed an optimization model with high execution time that increases rapidly as the number of EC increases. Furthermore, the proposed intra-EC VNE solution is performed in an aggregated fashion and does not consider any co-location criterion. In addition, the intra-EC part of the VNE problem is only considered as a constrained variable in the respective formulations in [99, 100], while the models contemplate the end-to-end SFC path embedding, which is not directly apply for hybrid-SFCs towards round-trip delay minimization. The same stands for the path-based solutions in [101, 102], where search-space reduction techniques are applied, but mainly links capacity constraints are taken into account, while in [103] the VNE is performed based on link and network switch modeling, with abstractly described computing constraints, in the absence of the execution order constraints, in accordance with the SFC specifications [46]. Zheng et al. [97] deals with the hybrid-SFC nature of

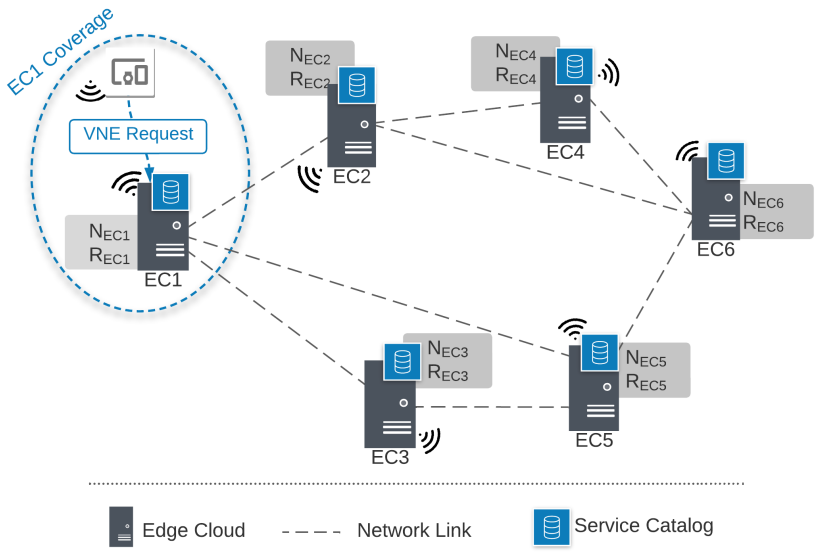


Figure 4.1: Edge Network Topology.

VNE, with round-trip delay minimization. However, the described solution focuses on constructing the DVNE path, regardless of how the service functions are placed within the ECs. The literature includes numerous variations of the both VNE and DVNE problems. The majority of the studies provide efficient solutions for end-to-end delay minimization in the VNE problem, under several other constraints. However, this dissertation focuses on providing distributed VNE solutions with the objective of minimizing the round-trip delay, in a time-efficient manner that will be possible to run online and deal with the dynamic nature of the modern 5G virtualized applications. The latter is motivated by the fact that reconfiguration of the embedding is frequently required due to end-device mobility or workload management and resource allocation constraints from the edge infrastructure perspective.

4.3 System Modelling

As described earlier, most of the works in the literature focus on the standard unidirectional SFC, where the traffic follows a path from source to destination, through the specified VNFs in accordance with their execution order [46]. However, in particular IoT scenarios, the application execution outcome is required to be returned to the end-device. For this type of SFC, the result is required to be transmitted back to the end-devices, and is defined as a hybrid-SFC [46] (hSFC). For hSFCs, a forward path to a specific destination point, which could be on the Cloud, for processing the forward traffic, and a backward path, through which the processing result is returned back to the end-device, are defined. These paths are not necessarily identical [97]. Also, the destination

VNF of an hSFC could not be specific or fixed, and so the forward and backward path compose a continuous data stream among the defined sequence of the VNFs [46]. Due to our focus on the deployment of virtualized applications at the Cloud Continuum, from Edge to Cloud, we extend the hSFC concept to a more holistically defined schema and we refer to that as a Virtual Network.

Definition 7. *A Virtual Network (VN) is a set of VNFs and virtual links, which is determined by a specific execution sequence, and the corresponding computing and network resource demands. The execution outcome is required to be returned to the source of the network traffic (end-device).*

The rest of this section describes the system modeling and the parameters of the DVNE problem in detail. Figure 4.1 illustrates the Edge Network topology, which consists of geographically distributed EC infrastructures. End-users can submit their VNE requests in any approximate EC. As mentioned above, in the context of Edge Computing, we assume that the services of a Virtual Network are deployed as VNFs in the form of SFC. A VNE request dictates the placement of the individual VNFs and includes both *(i)* new VNFs for deployment and *(ii)* shared VNFs that are already deployed in specific ECs.

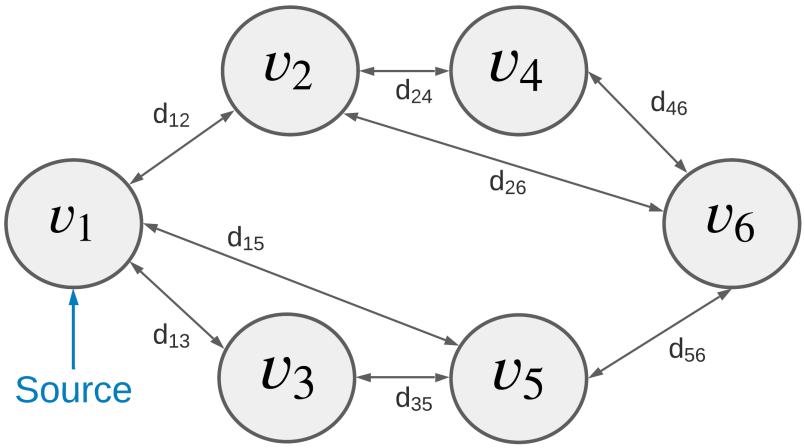


Figure 4.2: Edge Network's Graph Representation.

4.3.1 Substrate Network Model

We illustrate the Edge Network as an undirected Graph $G = (V, E)$. The graph representation of the Edge Network of Fig. 4.1 is illustrated in Fig. 4.2, while the set of network model parameters is shown in Table 4.2. In order to avoid confusion, we state that with the word *edge*, we refer to Edge

Computing artifacts, and with the word *link* to the network connection between the EC nodes or VN nodes, as well as the connections between the vertices of the respective graph representation of them. The set $V = \{v_1, \ldots, v_n\}$ corresponds to the nodes of G, where $v_i \in V$ represents the EC_i of the Edge Physical Network. The network links between the ECs v_i, v_j of the Edge Network constitute the set E, where $e_{ij} = (v_i, v_j) \in E$. The available computing resources, R_v, and the provided shared VNFs, $\mathcal{N}_v$, are determined as the main attributes of an EC $v \in V$. Besides that, d_{ij} denotes the delay of the network link e_{ij}. As it is shown in Fig. 4.1, the EC1 (v_1), in which the VNE request is initially submitted, is considered as the source node, denoted by v_0 and it is where the end-device is connected.

4.3.2 VNE Request Model

The end-user submits a VNE request in any EC $v \in V$. The end-device, which is the source of the network traffic, is denoted as s_0. Let $G_s = (V_s, E_s)$ be the graph representation of the VNE request. The set V_s denotes the VNFs that constitute the VNE request. Every element of V_s is a tuple $\langle s_i, rs_i \rangle$, where s_i represents the i^{th} VNF of the VN, and rs_i its resource requirements; as the shared VNFs within the VN are already deployed in an EC, it stands that if s_i is a shared-VNF then $rs_i = 0$. Furthermore, the E_s is the set that contains the virtual links between the VNFs of the request. Each virtual link $\big(s_i, s_j\big) \in E_s$ is associated with its residual bandwidth demand, denoted as b_{ij}. Figure 4.3 illustrates a VNE request, which consists of four VNFs. The s_3 is a shared-VNF and as we mentioned above, its resource requirements are considered equal to zero regarding the VNE. An example of an embedding solution for this VNE request on the aforementioned edge network is illustrated in Fig.4.4. As shown, the VNE request is submitted in the EC1, as the end-device operates in its coverage. For this VNE request, the VNFs s_1, s_2 are embedded in the EC2, the s_3 in the EC6, and the s_4 in the EC3. In this case, the EC1 does not host a VNF of the request; it is just considered as the ingress node of the solution, which includes the necessary paths for the VNE.

4.4 DVNE Problem Formulation

The discussed DVNE problem is broken down into two subproblems. The first one concerns an initial mapping of the VN, or parts of it, in all ECs of the Edge Network. Then, the construction of the VN is performed in a distributed manner, where each VNF $s \in V_s$ has to be deployed in one of the candidate ECs selected on the mapping of the first phase.

4.4.1 Initial Intra-EC VN mapping

Aiming at providing a DVNE solution, an initial mapping of VN parts in the respective EC infrastructures of the Edge Network is performed. With this capacity, each VNF will have a set of candidate ECs that could be embedded onto. Towards the initial mapping of a VN within the ECs,

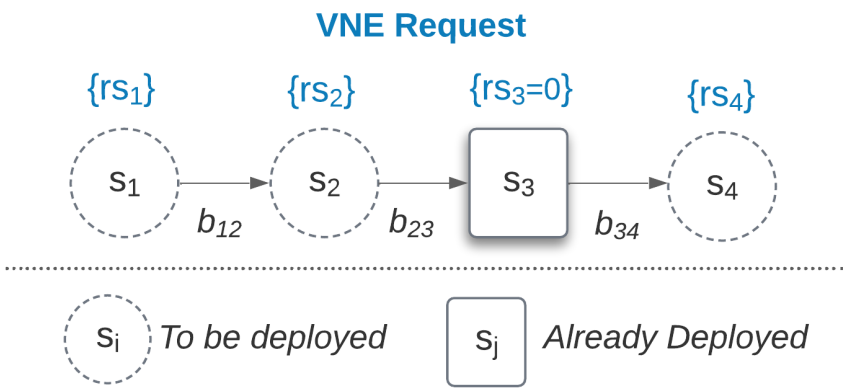


Figure 4.3: Virtual Network Embedding Request Model.

several parameters must be taken into consideration. In particular, shared-VNFs are pre-deployed in specific ECs, and the residual computing and network capacity of the ECs is constrained in terms of the available resources. Taking that into account, a primary fact is that a VNE request will have to be partitioned, initially, into more than one sub-VNs. In essence, this first phase is actually formulated as a typical SFC embedding problem in a cloud infrastructure [14]. In this work, we refer to this problem as intra-EC VN mapping. The goal of this initial phase is the minimization of computing and network resource utilization through optimized mapping of sub-VNs within an EC infrastructure. Moreover, several works in the literature highlight the importance of co-location between pairs of service nodes (VNFs) of a network slice embedding, which is another version of the DVNE problem, in a cloud data center [55], especially regarding the network latency and bandwidth, as well as other resource allocation parameters (CPU utilization, Inter-Rack network traffic within the EC infrastructure). Therefore, the main objective regarding the initial intra-EC VN mapping is to achieve a high co-location ratio between adjacent VNFs of a VNE request. Then, the initial mapping is used on the solution of the DVNE subproblem.

4.4.2 Delay-Aware Distributed VNE

The initial intra-EC VN mapping computes actually multiple embedding alternatives of entire (or part of) VNE request in various ECs. The goal of the second phase is to provide a DVNE solution that minimizes the round-trip delay. Assuming that an end-device generates the incoming traffic of the VN and the response of the corresponding services (VNFs) is returned to this device, we determine the *Virtual Network Path (VNP)* as a circuit with source (and sink) node the EC that the end-device is connected. Given a VNE request with $V_s = \{s_1, s_2, \ldots, s_m\}$ and the source of the network traffic s_0 (end-device), we define VNP as:

$$VNP = \{s_0, s_1, s_2, \ldots, s_m, s_0\}. \tag{4.1}$$

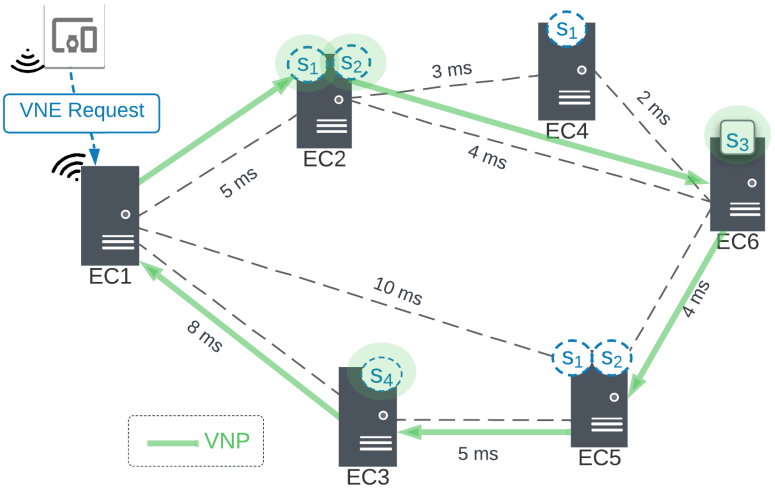


Figure 4.4: An example of Distributed VNE.

Towards a path-based DVNE formulation, the following definitions are required; *(i)* A DVNE solution is denoted by $\mathcal{P}$ and determines the sequence of the host ECs for every VNF of a VNE request, starting and ending in the EC v_0 that the end-device s_0 is connected,

$$\mathcal{P} = \{u_0, u_1, \ldots, u_m, u_{m+1}\}, \tag{4.2}$$

where $u_0 = u_{m+1} = v_0$, and u_i corresponds to the EC v that is the host of a VNF s_i in the solution $\mathcal{P}$, for $i = 1, \ldots, m$. So, $u_i = v \in V$. It is worth mentioning that the node v_0, which is mapped onto the variables u_0 and u_{m+1} of the solution $\mathcal{P}$, is the EC to which the end-device is connected and does not necessarily have to host a VNF of the request. Thus, it could be considered only as the first hop of the VNP, without the need to allocate resources. It stands that $\mathcal{P} \subset V$, while its cardinality is: $|\mathcal{P}| = m + 2$. The binary variable $\rho_i(v)$ indicates whether an EC v is a host for a VNF s_i in a DVNE solution $\mathcal{P}$,

$$\rho_i(v) = \begin{cases} 1, & \text{if } u_i = v \\ 0, & \text{otherwise} \end{cases}, \; i = 1, \ldots, \; m. \tag{4.3}$$

(ii) The cost of the shortest path between two ECs v_a, v_b in the graph representation G of the Edge Network, corresponds to the network delay between the respective ECs and is denoted by $w(v_a, v_b)$. The round-trip delay of a solution $\mathcal{P}$ is denoted by $D_{rt} = \sum_{i=0}^{m} w\big(u_i, u_{i+1}\big)$. Taking into account the aforementioned system modeling, our objective is to compute a DVNE solution that

Table 4.2: System's key notations.

Parameter	Definition
Edge Network Parameters	
G	Edge Physical Network graph
V	The set of EC nodes $v_1, v_2, \ldots$ of G
E	The set of links (v_i, v_j) of G
$\mathcal{N}_v$	Available shared VNFs in node $v \in V$
R_v	Available Computing Resources of node $v \in V$
d_{ij}	The network delay of the link $e(v_i, v_j) \in E$
VNE Request Parameters	
G_s	NSE request service graph
V_s	The set of VNFs in G_s
E_s	The set of virtual links (s_i, s_j)
s_i	The i^{th} VNF in set V_s
rs_i	The demanded resources of VNF s_i
b_{ij}	The bandwidth demand of link $(s_i, s_j) \in E_s$
Problem Formulation Parameters	
v_0	The EC that the end-device is connected
u_i	A host EC v for the VNF s_i, $i = 1, \ldots, m$
$\mathcal{P}$	A DVNE solution
$w(v_i, v_j)$	The shortest path's cost between v_i, v_j
D_{rt}	The round-trip delay of a DVNE solution $\mathcal{P}$

minimizes the round-trip delay D_{rt}:

$$\min_{P^*} \sum_{i=0}^{m} w\big(u_i, u_{i+1}\big) \tag{4.4a}$$

s.t.:

$$\sum_{v \in V} \rho_i(v) = 1, \ \forall i \in \{1, \ldots, m\} \tag{4.4b}$$

$$\sum_{i=1}^{m} \rho_i(v) rs_i \leq R_v, \ \forall v \in V \tag{4.4c}$$

$$w\big(u_i, u_{i+1}\big) < \infty, \ \forall i \in \{0, 1, \ldots, m\} \tag{4.4d}$$

The constraint (4.4b) indicates that for every solution $\mathcal{P}$, each service s_i has a unique hosting EC v, for a specific VNE request. The constraint (4.4c) guarantees that the requested resources by VNFs do not exceed the resource availability of every EC, while (4.4d) ensures the existence of a path between the ECs that compose the DVNE solution. In essence, (4.4d) focuses on the connectivity of the graph that corresponds to the solution P; the paths that are going to be used in the solution must exist, also, in the Edge Network Topology graph representation. In the following, the proposed solution is thoroughly described and consists of two parts: *(1)* the initial intra-EC

Virtual Network Mapping, which provides the (partial or not) mapping of the VNE request within the ECs of the Edge Network and guarantees the constraint (4.4c), and *(2)* the main shortest paths-based algorithm that undertakes the composition of a DVNE solution, based on the initial VN mapping, with minimized round-trip delay, while dealing with the exponential complexity of the problem, achieving a fast performance in terms of execution time.

4.5 Delay-Aware Distributed VNE Solution

4.5.1 Initial Intra-EC VN Mapping

For the initial VN mapping, various existing works rely on heuristic approaches to deal with the combinatorial nature of the problem, as the VNE is an *NP-Complete* problem [105]. Typical approximation algorithms for VNE concern *bin-packing problem* approaches, like *First-Fit algorithm*, for the initial VNF mapping of a VN within an EC infrastructure [106]. In this section, we propose a heuristic approach to undertake the initial mapping of the VNE request within the ECs. This algorithm associates every VNF of the VN with a set of candidates EC that can be hosted onto. In detail, the VNE request is processed for each EC, and the mapping that occurs for every VNF of the VN would be accepted by the corresponding EC, during the next phase of the distributed solution construction. It is worth mentioning, that a VN could be mapped onto an EC partially, depending on resource availability and other constraints, as these are described below. Aiming to minimize the servers' utilization and the bandwidth consumption within an EC infrastructure, while achieving a notable VNF pair co-location ratio, our approach leverages the fact that a VNE request includes pre-deployed shared-VNFs on specific servers of an EC, in order to provide a mapping solution.

We assume that the set $\Xi = \{\xi_1, \ldots, \xi_K\}$ contains all the active servers of an EC infrastructure. A server $\xi \in \Xi$ has an available computing capacity c_ξ. Furthermore, the physical links between the servers via a switch are defined by the set $\mathcal{L} = \{l_\xi \mid \xi \in \Xi\}$ and every link is associated with a certain available bandwidth value β_{l_ξ}. As we already mentioned, the shared-VNFs are already deployed and hosted on specific EC servers. We assume that the set $SH = \{s_i | s_i \in V_s : rs_i = 0\}$, contains the shared-VNFs of the request, where $i = 1, \ldots, \mu$. We also define the function $h(s, \xi)$, to express whether a server ξ hosts the VNF $s \in V_s$,

$$h(s,\xi) = \begin{cases} 1, & \text{if } \xi \text{ is the host of } s \\ 0, & \text{otherwise} \end{cases}. \tag{4.5}$$

The mapping of a VNF s_i in a server ξ has to satisfy the following constraints regarding the computing capacity demand rs_i and the bandwidth demand $b_{i,i+1}$, between adjacent VNFs.

(1) Computing capacity constraint: This constraint ensures the mapping feasibility based on the aforementioned policy. Particularly, for shared-VNFs examines if the candidate server ξ is the corresponding host, while for VNFs to be deployed scrutinizes the available capacity in the server

with respect to the VNF demand. These define the following parameter α_1:

$$\alpha_1 = \begin{cases} 1, & \text{if } h(s_1, \xi) = 1 \ \& \ s_i \in SH \text{ (deployed in } \xi) \\ 1, & \text{if } rs_i \leq c_\xi \ \& \ s_i \notin SH \text{ (to be deployed in } \xi) \\ 0, & \text{otherwise} \end{cases} . \tag{4.6}$$

(2) Bandwidth capacity constraint: The bandwidth of the virtual link (s_i, s_{i+1}), $b_{i,i+1}$ is allocated by the mapping of the VNF s_{i+1}, as for each partition the mapping sequence is reversed. Although, if the s_i is co-located with the s_{i+1}, the previous bandwidth demand is released, and the next virtual link demand is allocated, which is $b_{i-1,i}$. Thus, the following parameter is defined to express the bandwidth constraint of a VNF s_i mapping in the candidate server ξ:

$$\alpha_2 = \begin{cases} 1, & \text{if } h(s_{i+1}, \xi){=}1 \text{ (same hosts)} \\ 1, & \text{if } b_{i,i+1} \leq \beta_{l_\xi} \ \& \ h(s_{i+1}, \xi){=}0 \text{ (different hosts)} \\ 0, & \text{otherwise} \end{cases} . \tag{4.7}$$

where, for s_i has to be mapped onto server $\xi \in \Xi$, if $h(s_{i-1}, \xi) = 0$, the demanded bandwidth of the virtual link have to be available in the physical link l_ξ, while in the case where $h(s_{i-1}, \xi) = 1$, which means that the adjacent VNFs are co-located and no bandwidth is consumed from the corresponding physical link l_ξ.

Algorithm 2 includes the steps toward the intra-EC VN mapping. Initially, between the lines 1 and 12 the VN partition phase takes place. The partitioning of the VNE request is performed, based on the position of the shared-VNFs in the VN, where each partition is a path of VNFs, starting from a new VNF for deployment and reaches the shared-VNF as a destination on the partition path (line 1). Assuming μ shared-VNFs in V_s, we denote $\mathcal{U} = \{V_s^1, \ldots, V_s^\mu\}$ as the set that contains the partitions of V_s, where $V_s^1 \cup V_s^2 \cup \ldots \cup V_s^\mu = V_s$. Each partition mapping begins from the host server of the shared-VNF, while the rest available servers are sorted in ascending order (Ξ') on their available resources (line 3). Between the partitions of the same VN, a *Best-Fit* policy is adopted, meaning that the next partition that has to be mapped will start its VNF mapping from its corresponding shared-VNF (line 5), while next on the list of candidates servers will follow the hosts of the previous partition. The mapping of a partition is undertaken by the PARTITIONMAPPING() procedure (line 7). If a partition is not able to be mapped due to unavailable resources within the EC, it will be further partitioned according to its resource demands (line 9), and the mapping procedure will be invoked again (line 10). The mapping of a VN partition is performed by the PARTITIONMAPPING() procedure (lines 13-31). The mapping process begins from the host server of the shared-VNF, if the latter is hosted on this EC, and following the partition path backward (lines 14-15), the remaining VNFs are placed in a *Worst-Fit* fashion based on their resource demands (lines 16 - 29). In line 19, the mapping constraints are checked for each VNF of the partition. For enhanced VNF pair co-location and reduced inter-server network traffic, at each

Algorithm 2 Intra-EC VN Mapping Heuristic

Input: G_s, $\mathcal{N}_v$, Ξ, $\mathcal{L}$, SH
Output: A mapping $\mathcal{H}$ of VNFs of a VNE request

1: $\mathcal{U} \leftarrow \{V_s^1, \ldots, V_s^\mu\}$
2: **for** $V_s^i \in \mathcal{U}$ **do**
3: $\Xi' \leftarrow$ sort Ξ based on c_ξ in reverse order
4: **if** $i > 0$ **then**
5: Bring host of $s_{i-1} \in SH$ on top of Ξ'
6: **end if**
7: mapped $\leftarrow$ PARTITIONMAPPING(V_s^i, Ξ', $\mathcal{L}$)
8: **if** not mapped **then**
9: re-partition V_s^i based on resource requirements
10: mapped $\leftarrow$ REPARTITIONMAPPING(V_s^i, Ξ', $\mathcal{L}$, 1)
11: **end if**
12: **end for**
13: **procedure** PARTITIONMAPPING(V_s^i, Ξ', $\mathcal{L}$)
14: Bring host of $s_i \in SH$ on top of Ξ' if $s_i \in \mathcal{N}_v$
15: Reverse the order of the V_s^i
16: **for** $s_j \in V_s^i$ **do**
17: check $\leftarrow False$
18: **for** $\xi \in \Xi'$ **do**
19: **if** (4.6) and (4.7) satisfied **then**
20: check $\leftarrow True$
21: Map s_j on server ξ
22: Update values of c_ξ and β_{l_ξ}
23: Bring ξ on top of Ξ'
24: **end if**
25: **end for**
26: **if** not check **then**
27: **return** $False$
28: **end if**
29: **end for**
30: **return** $True$
31: **end procedure**
32: **procedure** REPARTITIONMAPPING(V_s^i, Ξ', $\mathcal{L}$, δ)
33: **if** $\delta = |V_s^i|$ **then**
34: **return** $False$
35: **end if**
36: $newParts \leftarrow |V_s^i| - \delta$ length parts of V_s^i
37: Sort $newParts$ in descending order based on resource demands
38: **for** $V_s^{i*} \in newParts$ **do**
39: **if** PARTITIONMAPPING(V_s^{i*}, Ξ', $\mathcal{L}$) **then**
40: **return** $True$
41: **end if**
42: **end for**
43: $\delta \leftarrow \delta + 1$
44: REPARTITIONMAPPING(V_s^i, Ξ', $\mathcal{L}$, δ)
45: **end procedure**

iteration, the host of the last placed VNF, emerges at the top of the list of candidates to host the next VNF (lines 20 - 23). The *check* parameter identifies when a partition is not able to fit within the EC, while it returns $False$ (lines 26-27) as a result of the PARTITIONMAPPING() procedure and invokes the re-partitioning. If a partition of length $|V_s|$ does not fit in the EC servers, the heuristic selects sequences of VNFs of length $|V_s| - 1$ and maps one of them, starting from the one with the maximum resource requirements. If none of the $|V_s| - 1$ length partitions is mapped, the sequence of VNFs of length $|V_s| - 2$ is considered and so on. The process continues until no VNF remains after re-partitioning. The re-partition policy aims to maximize the number of adjacent VNFs that can be co-located during the mapping. The parameter δ determines the number of sequential VNFs to be considered for the $|V_s| - \delta$ partitions. These partitions are sorted in descending order based on

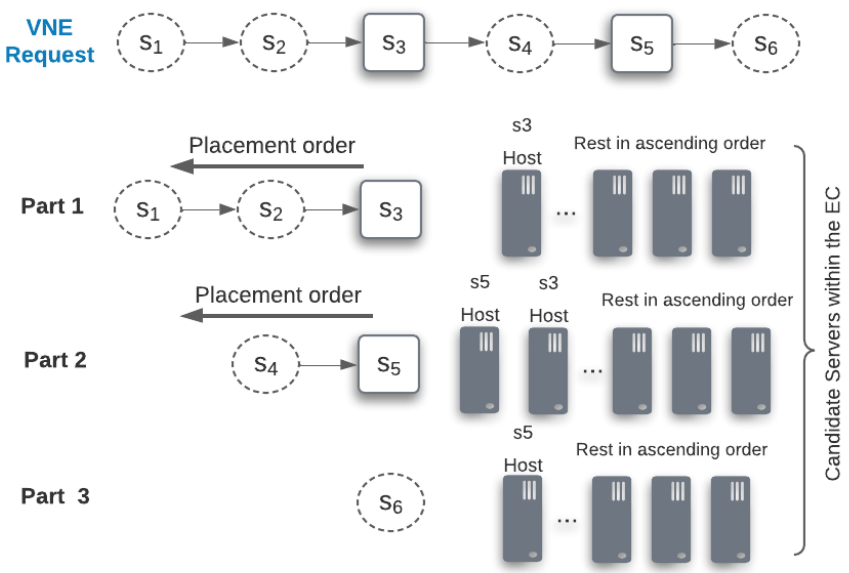


Figure 4.5: The proposed heuristic for intra-EC VN Mapping.

the total resource requirements. Then, the algorithm iterates through these partitions and maps the first that fits within the servers Ξ' by invoking the basic PARTITIONMAPPING() procedure.

Figure 4.5 illustrates an example, where a VNE request consists of two shared-VNFs from a total of six. Following the above described policy, the VN is partitioned based on the position of the shared-VNFs in three distinct partitions. For the sake of demonstration, the mapping process for the first partition $V_s^1 = \{s_1, s_2, s_3\}$ is performed as follows: The mapping attempt starting backwards from the VNF s_3. As s_3 is a shared-VNF, it is already placed in a server, which is its host, denoted by $\xi_a \in \Xi$, and $h(s_3, \xi_a) = 1$. The ξ_a is now the first candidate in the list of servers Ξ' for the VNF s_2 to be mapped, based on resource constraints, aiming to minimize the utilised servers and the inter-server network traffic; otherwise the *Worst-Fit* is performed. Then, for the s_1 VNF, the first candidate server in the list Ξ' would be the host of s_2.

The output of the algorithm 2 is a mapping $\mathcal{H} = \{\eta_i\}_i$, where $i = 1, \ldots, m$. In specific, η_i denotes a server $\xi \in \Xi$ that hosts the VNF $s_i \in V_s$, while in the case where the VNF s_i has no host in the EC, it stands that $\eta_i = None$. Thus, EC can be a candidate host for the VNFs that are mapped as occurs for $\mathcal{H}$. The mapping $\mathcal{H}$ will be used below, from the algorithm that undertakes the distributed embedding of the VN.

4.5.2 Distributed VNE with Minimum Round-Trip Delay

The methodology for solving the DVNE problem of subsection 4.4.2 is presented here. Towards the minimization of the round-trip delay, we implement a shortest path-based approach that relies on

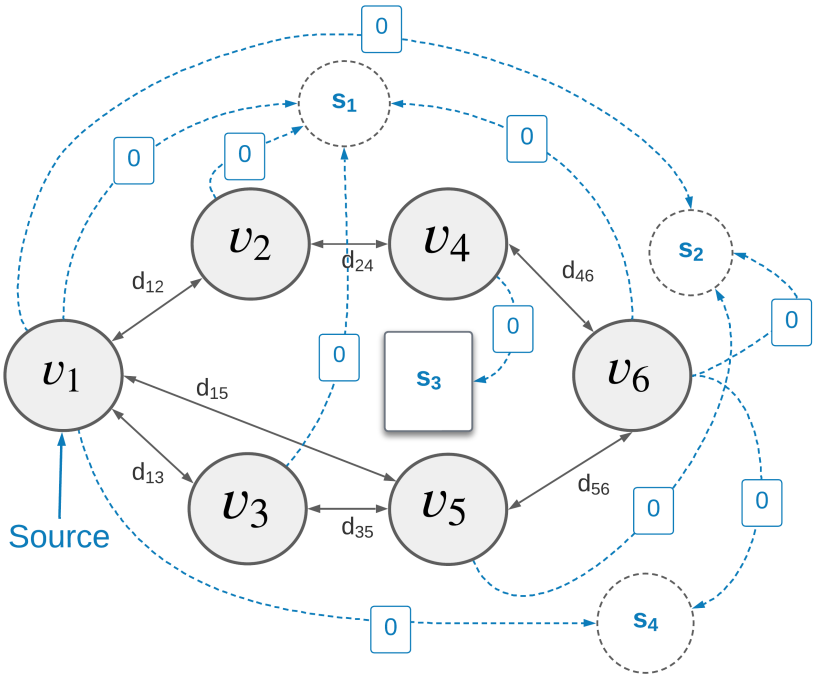


Figure 4.6: An augmented graph $\mathcal{G}_a$ based on G and G_s.

the Yen's *k-shortest-paths* algorithm [59], which calculates the k shortest loopless paths between two nodes in a graph (weighted or not), by utilizing the *Dijkstra's shortest-path algorithm*. As mentioned, in typical SFC embedding problems, a distinct VNF is the destination of the execution sequence. Imitating this approach, the proposed algorithm, namely *kSP-DVNE*, generates paths considering that every VNF $s \in V_s$ could be the destination of a VNP and for every potential destination, a forward VNP (f-VNP) and a backward VNP (b-VNP) are determined. For example, taking into consideration the VNE request in Fig. 4.3, assuming that the destination is the VNF s_2, the corresponding f-VNP is $\{s_0, s_1, s_2\}$, while the b-VNP is $\{s_2, s_3, s_4, s_0\}$. Additionally, a Virtual Network Embedding Path (VNEP) is defined for both the f-VNP and b-VNP, denoted as f-VNEP and b-VNEP respectively. These embedding paths determine the visited EC nodes that host the corresponding VNFs of f-VNP and b-VNP. The {f-VNEP}$\cup${b-VNEP} provides a DNVE solution $\mathcal{P}$, which is actually a circuit on the graph G, as an EC is able to be visited multiple times as it could host various VNFs of the VNE request. For the above example, the determined embedding paths are: f-VNEP= $\{u_0, u_1, u_2\}$ and b-VNEP= $\{u_2, u_3, u_4, u_0\}$. Below, a description regarding the main components of the proposed solution, is given.

4.5.2.1 Augmented Graph Construction

The initial intra-EC VN mapping solution provides the candidate ECs for hosting every VNF $s \in V_s$. This mapping is denoted by $\mathcal{H}_v$, $\forall v \in V$. The implementation of the proposed DVNE method relies on an *augmented graph* $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$, where the graphs G and G_s that represent the edge network and the VNE request are merged. In specific, the set $\mathcal{V}_a = V \cup V_s$ represents the graph's vertices and consists of these that correspond to the EC nodes V plus the VNFs nodes V_s. Furthermore, the set $\mathcal{E}_a = E \cup \{(v, s, 0) \mid \forall v \in V \text{ and } \forall s \in V_s \cap \mathcal{H}_v\}$ represents the augmented graph's edges. The latter consists of the edges of the initial substrate network links E, while for every EC v that is a candidate host for VNF s, an edge of zero weight $(v, s, 0)$ is added. For example, we consider the substrate network graph model of Fig. 4.2 and the VNE request of Fig. 4.3. An example of an augmented graph $\mathcal{G}_a$ is depicted in Fig. 4.6. In this example, we assume that, depending on the mapping of each EC, the VNF s_1 can be hosted $\{v_1, v_2, v_3, v_6\}$. So, $\mathcal{E}_a$ contains all the edges of the set E, while adding to it the edges $(v_1, s_1, 0), (v_2, s_1, 0), (v_3, s_1, 0)$ *and* $(v_6, s_1, 0)$. Similarly, edges from candidate ECs for hosting the remaining VNFs are accordingly added in the graph $\mathcal{G}_a$. The construction of the augmented graph is essential for the development of the proposed solution towards the round-trip delay minimization. It is worth mentioning that the network delay between two co-located VNFs in the same EC is considered negligible and is determined equal to zero.

4.5.2.2 kSP-DVNE algorithm

Regarding DVNE, the proposed approach takes into account the characteristics of the typical SFC embedding problem. As mentioned above, a VNE request is submitted in a specific EC. The VNFs that comprise the VN, can be hosted in various ECs of the Edge Network. Some of the VNFs of the VNE request have to be embedded with respect to resource constraints, while the shared-VNFs are already deployed on specific ECs. Also, the execution outcome of the VN has to be directed back to the source; that is the end-device, which generated the request. This implies no specific destination EC for the network traffic, neither a predefined forward and backward service function path. To deal with this abstract nature of the VN, our approach aims to examine various instances of the VN, with different destination VNF (and a corresponding host EC) following different paths towards the respective destination VNF per case. Obviously, a VNF s could have several candidates hosting ECs v, and various paths from the source to the destination EC v. The number of the possible different destinations VNFs of a VN equals to the VN's length, m. Consequently, m will also be the number of the distinct f-VNPs and b-VNPs. Aiming to provide the DVNE solution with minimized round-trip delay, the proposed approach is based on the following remark:

Remark 1. *Let* $\mathcal{SP} = \{p_{sh}(s_1), p_{sh}(s_2), \ldots, p_{sh}(s_m)\}$ *the shortest paths from a source* v_0 *to every VNF* $s_i \in V_s$ *of a VN in* $\mathcal{G}_a$. *If* p_{sh}^{max} *is the maximum cost path in* $\mathcal{SP}$ *and* s_{dst}^{sh} *the corresponding*

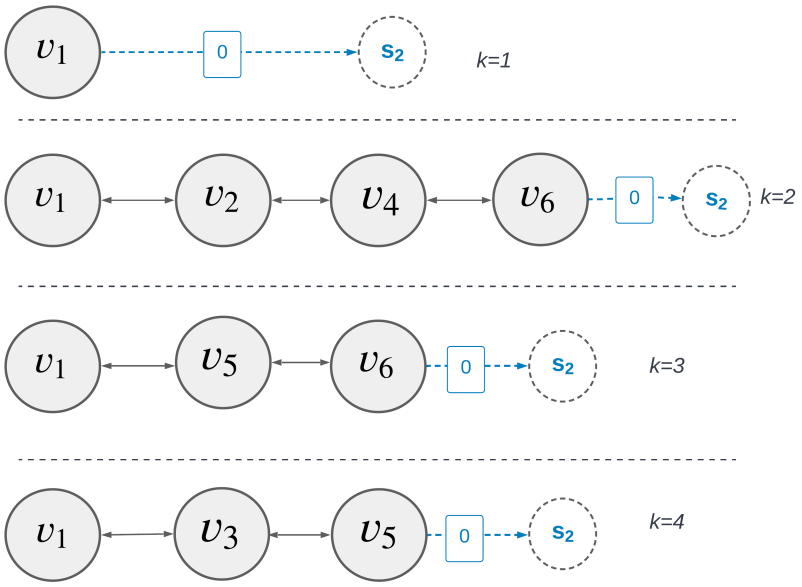


Figure 4.7: An example of four possible shortest paths from s_0 to s_2 in $\mathcal{G}_a$.

destination VNF, the minimum round-trip delay of a DVNE is:

$$D^*_{rt} \geq 2 \times w(v_0, s^{sh}_{dst}).$$

Given the fact that the network traffic has to traverse every host u_i of each VNF s_i, in the best case, the shortest reachable host for a feasible DVNE solution, is the one corresponding to p^{max}_{sh}. This path corresponds to a destination VNF s^{sh}_{dst}. If the f-VNP and the b-VNP can be embedded in the p^{max}_{sh}, in a way that every edge is traversed at most once from the f-VNEP and b-VNEP individually, the round-trip delay D^*_{rt} of this embedding solution is the minimum achievable. Therefore, the main goal of this approach is to generate multiple paths that could be examined in a similar way; that is to find a f-VNEP and a b-VNEP on the same path, while the path's edges are traversed only once for the corresponding f-VNEP and b-VNEP. Below, more details regarding the proposed method are given.

Candidate Embedding Paths: Towards the generation of a sufficient number of paths for every destination VNF, we rely on the Yen's *k-shortest paths (k-SP)* algorithm [59], which provides the $k \in \mathbb{N}$ shortest loopless paths between two vertices in a graph. Given a VNE request of m VNFs and a source EC v_0, we consider the m possible destination VNFs $s_{dst} \in V_s$. The *candidate embedding paths* are generated by computing k shortest paths from u_0 to each VNF s_{dst}. We define the k shortest paths between the source EC and a VNF s as:

$$P^k_s = \{p^1(s), p^2(s), \ldots, p^k(s)\}, \tag{4.8}$$

Algorithm 3 Delay Aware kSP-DVNE Algorithm

Input: $\mathcal{G}_a,\ G,\ G_s$
Output: $\mathcal{P}^*,\ D^*_{rt}$

1: **for** $s \in V_s$ **do**
2: $P^k_s \leftarrow$ `k-Shortest Paths` from u_0 in $\mathcal{G}_a$
3: **end for**
4: $\mathcal{C} \leftarrow \{P^k_s\},\ \forall s \in V_s$
5: **for** $p_c \in \mathcal{C}$ **do**
6: **if** (4.10) is satisfied **then**
7: Identify the destination VNF s_{dst} in p_c
8: f-VNP $\leftarrow \{s_0, \ldots, s_{dst}\}$
9: b-VNP $\leftarrow \{s_{dst}, \ldots, s_0\}$
 # Embed f-VNP on p_c (reverse order)
10: f-VNEP $\leftarrow \{u_0, u_1, \ldots, u_{dst}\}$
 # Embed b-VNP on p_c
11: b-VNEP $\leftarrow \{u_{dst}, \ldots, u_m, u_0\}$
12: Construct the embedding solution
13: $\mathcal{P} \leftarrow$f-VNEP $\cup$ b-VNEP
14: **if** (4.4b) satisfied **then**
15: $D_{rt} \leftarrow \sum_{i=0}^{m} w(u_i, u_{i+1}),\ u_i \in \mathcal{P}$
16: **if** (4.4d) satisfied **then**
17: Check if $D_{rt} < D^*_{rt}$
18: Update D^*_{rt} and $\mathcal{P}^*$ if needed
19: **end if**
20: **end if**
21: **end if**
22: **end for**

where $p^i(s)$ is the i^{th} shortest path from v_0 to s in $\mathcal{G}_a$ and contains the traversed ECs $v \in V$ with the sequence that they are visited. Thus, the candidate embedding paths constitute the set:

$$\mathcal{C} = \bigcup_{s \in V_s} P^k_s. \tag{4.9}$$

As m is the number of different destination VNFs, also m distinct pairs of f-VNPs and b-VNPs are determined. Subsequently, the paths of $\mathcal{C}$ will be examined for efficient DVNE solutions, in terms of round-trip delay.

kSP-DVNE algorithm: Algorithm 3 describes the steps to provide a solution for the DVNE problem. To begin with, the k shortest paths in the augmented graph $\mathcal{G}_a$ are computed, from the source EC v_0 to each vertex that corresponds to a VNF of the VN (lines 1-3), as described previously. Then, each candidate path $p_c \in \mathcal{C}$ is examined (line 6) based on the following policy: A candidate path p_c must contain at least one host EC for every VNF $s \in V_s$. Given the augmented graph $\mathcal{G}_a$, a candidate path $p_c \in \mathcal{C}$ is able to provide a DVNE solution if the following condition is satisfied:

$$\forall s \in V_s, \exists v \in p_c : \exists (v, s) = 0, \ where \ (v, s) \in \mathcal{E}_a. \tag{4.10}$$

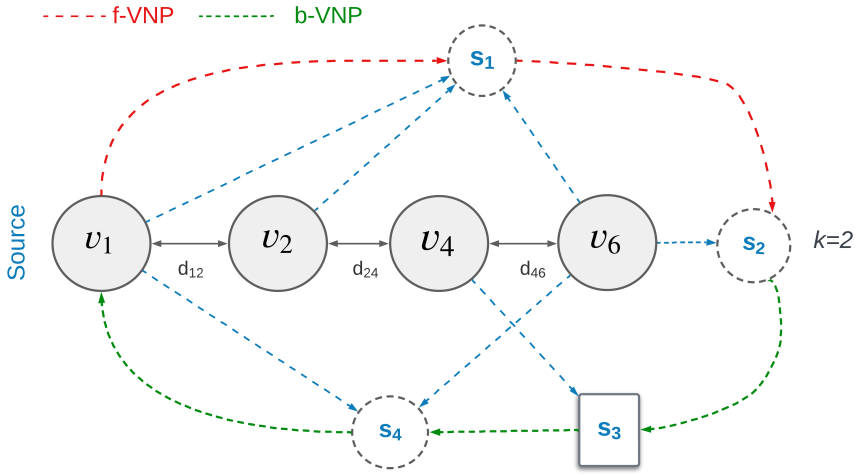
In essence, an edge between an EC v and a VNF s must exist in $\mathcal{G}_a$, so, the path $p_c \in \mathcal{C}$ is a *valid candidate* path.

f-VNP and b-VNP Embedding: Subsequently, for a valid candidate path p_c, an embedding solution is constructed. More precisely, as described above, the solution construction can be divided into the f-VNP and b-VNP embedding, which leads to the f-VNEP and b-VNEP, respectively (lines 7-11). Regarding the f-VNP $= \{s_1, \ldots, s_{dst}\}$, the process starts from the s_{dst} host, which is the last EC vertex of p_c. Accessing the f-VNP in reverse order, (*i.e.,* $s_{dst-1}, \ldots, s_1$), we seek for the first available host of the next VNF, following the p_c vertices, also, in the opposite direction. For the b-VNP case, we access the VNFs straight forward (*i.e.,* $s_{dst+1}, \ldots, s_m$), while we seek for the first available host of the next VNF, following the p_c vertices in the direction towards the source EC. The DVNE solution is defined based on the f-VNEP and b-VNEP, its round-trip delay is computed (lines 12-18). Figure 4.8 illustrates an example of the embedding solution construction, considering the path $p^2(s_2)$ of Fig. 4.7. The destination VNF s_2 is placed in the corresponding host EC of the candidate path v_6. The EC v_1 is the source EC, and so $v_0 = v_1$, while the f-VNP is $\{s_0, s_1, s_2\}$ and the b-VNP is $\{s_2, s_3, s_4, s_0\}$. Regarding the end-device, which is the source s_0 of the network traffic, the EC that it is connected is the $v_0 = v_1$. Starting the embedding from f-VNP, the s_2 is hosted on v_6. The next VNF to be placed is the s_1. The first EC that is a candidate host for s_1 from v_6 towards v_1 will be chosen. In this case, v_6 is nominated as a host of s_1. The f-VNEP is then $\{v_1, v_6, v_6\}$, containing the host ECs for the corresponding services. Concerning the b-VNP embedding, the shared VNF s_3 has only one candidate host EC v_4, where it is already placed. Then, for the s_4, exclusively the ECs after the v_4 and backwards (toward v_0) are taken into account for identification of a candidate host, where v_1 is a the only choice for s_4. In this way, the b-VNEP is shaped as $\{v_6, v_4, v_1, v_1\}$. The construction of the final solution is the circuit starting from the $v_0 = v_1$ and traversing the respective hosts of each VNF of the request, observing the execution order in which they are defined. Thereafter, the DVNE solution is,

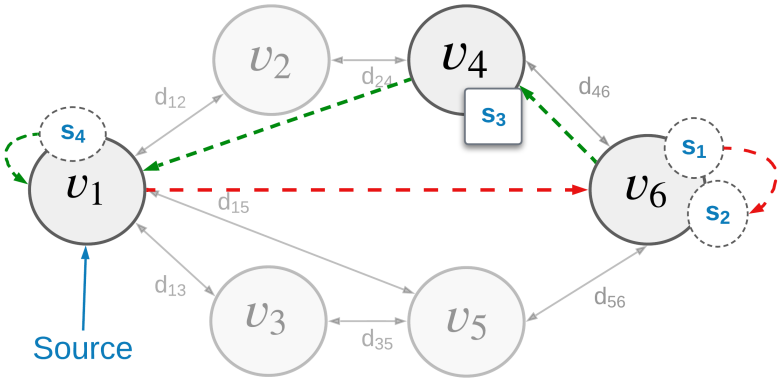
$$\mathcal{P} = \{u_0, u_1, \ldots, u_4, u_5\} = \{v_1, v_6, v_6, v_4, v_1, v_1\},$$

as $v_0 = v_1$. The round-trip delay of this solution is calculated using the shortest paths between the traversed ECs and equals to:

$$D_{rt}(\mathcal{P}) = \sum_{i=0}^{4} w(u_i, u_{i+1}).$$



(a) A candidate embedding path of a VN with destination VNF s_2.



(b) DVNE solution provided by the proposed kSP-DVNE heuristic.

Figure 4.8: A DVNE solution in the $p^2(s_2)$ path of Fig. 4.7.

4.5.2.3 Complexity Analysis

The proposed DVNE solution can be divided into three major parts: (i) the augmented graph $\mathcal{G}_a$ construction, (ii) the k-shortest paths set $\mathcal{C} = \{P_k\}$ calculation and (iii) the computation of the solution with the lowest round-trip delay $D_{rt}, \mathcal{P}$. From this perspective, we can analyze the complexity of the solution as follows:
(i) The construction of $\mathcal{G}_a = (\mathcal{V}, \mathcal{E})$ is based on $G = (V, E)$ and $G_s = (V_s, E_s)$. Every vertex $v \in V$ is examined as a candidate host for a subset of the VNFs $s_i \subset V_s$. So, for each $v \in V$, we iterate the VNFs which belongs to the mapping $\mathcal{H}_v$ and the corresponding edges of zero weight are added in the graph accordingly. Let $n = |V|$ and $m = |V_s|$, the augmented graph construction demands $n \times |\mathcal{H}_v|$ operations and, as $|\mathcal{H}_v| \leq |V_s| = m$, $\forall v \in V$, (at most m VNFs can be hosted in an EC), the time complexity is $\mathcal{O}(nm)$.
(ii) According to Algorithm 3, the following step is to compute the k-shortest paths, in order to create the set of the candidate paths to be examined. To achieve this, we utilize the Yen's algorithm in the graph $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$. As it described in [59], the time complexity of the algorithm $\mathcal{O}(k|\mathcal{V}_a|^3)$. With respect to our modeling, it is valid that $\mathcal{V}_a = V \cup V_s$, therefore, $|\mathcal{V}_a| = |V| + |V_s| = n + m$. Therefore, the time complexity of that algorithmic step is $\mathcal{O}(k(n+m)^3)$.
(iii) In order to provide the solution, the shortest path cost w that represents the network delay between every pair of nodes in V has to be calculated in prior for each VNE request. As described above, the round-trip delay of an embedding solution is $D_{rt} = \sum_{i=0}^{m} w\big(u_i, u_{i+1}\big)$. Using the Floyd-Warshall all pair shortest paths algorithm [107], we obtain the desired results in the complexity of $\mathcal{O}(n^3)$.
(iv) The candidate paths list is accessed just once in order to find the DVNE solution, and its length is equal to km. In an iteration, each path is validated if contains hosts for all services. At most, a path consists of n vertices that will have to be examined, as we care about vertices that represent ECs. Furthermore, two more times, the path is traversed for providing the mapping of the f-VNP and b-VNP, thus, $2n$ at most vertices could be accessed. The round-trip delay calculation is the sum of the pairs of vertices in $\mathcal{P}$, which have cardinality equal to $m+2$. Hence, the time complexity of the loop that calculates the DVNE solution is:

$$\mathcal{O}(km(n + 2n + (m + 2))) = \mathcal{O}\big(km(n + m)\big).$$

In summary, the complexity of the proposed algorithm is:

$$\mathcal{O}(nm + k(n+m)^3 + n^3 + km(n+m)) = \mathcal{O}(kn^3), \tag{4.11}$$

as $m << n$, where $m, n \in \mathbb{N}$. It is obvious that the algorithm's complexity depends on the k value, which determines the number of the candidate paths for the DVNE. A higher value of k implies larger search space for the algorithm and increased chances for finding a near optimal (or the optimal) embedding solution. Additional remarks regarding the effect of k parameter in the complexity of the algorithm are presented in the evaluation section. Next, we present

the performance evaluation of both the initial intra-EC mapping heuristic and the kSP-DVNE algorithm through simulation and comparison with relative studies.

4.6 Evaluation

In this section, the evaluation of the proposed approach that deals with the DVNE problem is presented in two distinct parts. Initially, the heuristic intra-EC VN mapping performance is analyzed and compared with a relative approach in the literature [106]. Subsequently, the Distributed VNE solution is evaluated and furthermore compared with an existing work regarding delay-aware hSFC embedding in a distributed manner at the network edge [97]. For every experiment, the impact of major parameters on the DVNE solution is illustrated through numerical results and relevant discussions. The implementation of both sV-VNM and kSP-DVNE approaches, as well as the overall simulation environment was performed using Python programming language. The same stands for the relevant approaches from the literature that are utilized to perform the comparisons with the algorithms that we propose in this paper. The ECs and the VN are implemented as two separate objects containing the computing and network attributes of the aforementioned system modeling. To generate the graph-based models of the Edge Network and the VNE request, the Python package *NetworkX* [108] was utilized. This package enables the corresponding parameters of the model to be parsed on the package's graph objects, as nodes' and links' attributes. The simulations took place on a Virtual Machine with installed operating system *Ubuntu 20.04*, with 4 virtual CPU cores and 16GB of RAM.

4.6.1 Intra-EC VN Mapping Heuristic Evaluation

Regarding the proposed heuristic for the intra-EC VN mapping, namely *shared VNF-based VN Mapping Heuristic (sV-VNM)*, the evaluation is performed through modeling and simulation, while as mentioned before a comparison with a similar approach [106] is also provided. This approach, namely the First-Fit Allocation heuristic (FFAh) allocates the VNFs of a SFC in a group of servers, which operate under the same core switch in an Edge Network. Similarly to sV-VNM, the FFAh aims at providing an initial VNF mapping towards the minimization of the power consumption (i.e., number of activated servers) within the EC and the minimization of the aggregated network traffic that the VNFs inject between the servers. Furthermore, this work leverages the VNF co-location to achieve its objectives. The main idea of the FFAh can be described as follows. For every cluster of servers, which compose an EC in our case, the FFAh sorts the servers in decreasing order with respect to their available resources. Afterwards, the FFAh strives to allocate each VNF of the SFC in a first-fit fashion. The above remarks make the FFAh a suitable alternative for comparison with the proposed sV-VNM approach.

Table 4.3: sV-VNM Evaluation - Simulation Parameters.

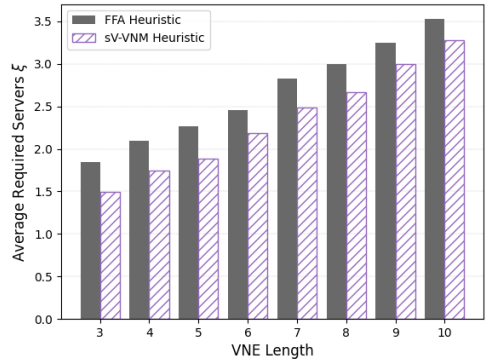
Parameter	**Value**
Number of ECs $\|V\|$	100
Number of active servers $\|\Xi\|$ per EC	$U[5, 10]$
VNE Length $\|V_s\|$ per request	$U[3, 10]$
Shared-VNFs in every request	$20\% - 30\%$ of $\|V_s\|$
Available cores per server	$U[2, 16]$
Available bandwidth per link	1 Gbps
Demanded cores per VNF	$U[1, 5]$
Demanded Bandwidth per virtual link	$U[30, 150]$
Total VNE requests	1000

4.6.1.1 Simulation Setup

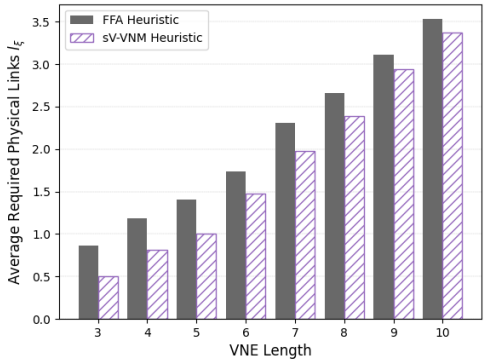
With respect to our system modeling, the EC is the corresponding cloud data center and it contains a group of available servers linked with the corresponding switch of the EC. The number of ECs equals 100, while the number of active servers $\Xi = \{\xi_i\}_i$ within each of them varies in the discrete uniform space $[5, 10]$. Regarding the computing and network resources, the available cores for each server, c_ξ, ranges from 4 to 16, while the available bandwidth capacity β_{l_ξ} of each l_ξ link, is set to 1 Gbps. Furthermore, we consider a pool of 20 distinct types of VNFs that a subset can be selected from, in order to compose a VNE request. Concerning the VNE request, its length ranges from 3 to 10, while up to 3 of these VNFs are shared-VNFs. For the rest VNFs, the demanded CPU cores are uniformly distributed in the space $[1, 5]$. The bandwidth demand of the virtual links, between adjacent VNFs ranges from 30 to 150 Mbps. In order to ensure fairness in comparative results regarding the performance of the considered heuristics, we assume that all the demanded shared VNFs of each VNE request are available in any of the ECs during the experiment. Each VNE request is submitted in any of the ECs of the substrate network, which, as mentioned above, will have different resource availability, and different placement of the shared VNFs. In total, 1000 VNE requests are generated and forwarded to all of the ECs. Table 4.3 includes all simulation parameters of the intra-EC VN Mapping experiment. The comparison of the two heuristics is based on the following metrics: (i) The average number of servers used per VNE request in every EC, (ii) the average number of physical links that are utilized (injected traffic between servers) in every EC, (iii) the percentage of the co-located on the same server adjacent VNFs and (iv) the percentage of pair co-location involving shared-VNFs with respect to the length of the VNE request.

4.6.1.2 sV-VNM and FFAh Comparative Results

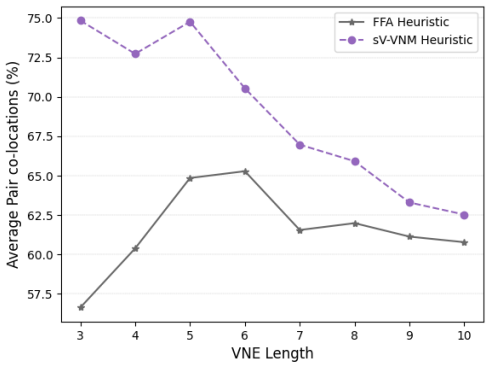
The numerical results are illustrated in Fig. 4.9. Regarding the average number of used servers, the sV-VNM utilizes 12-20% less servers regarding VNE of small and medium lengths (3-6 VNFs in the VNE) compared to the FFAh, while it still performs slightly better for larger VNE requests



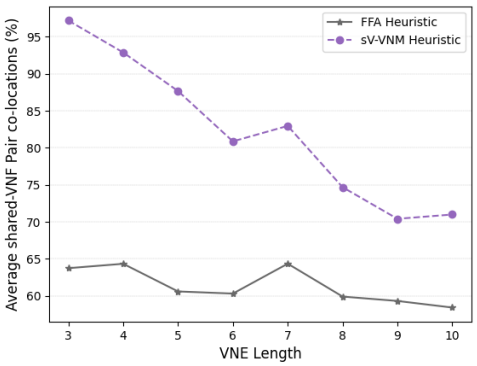
(a) Average number of utilized servers per EC.



(b) Average number of utilized links per EC.



(c) The pair co-location ratio between adjacent VNFs.



(d) The pair co-location ratio of adjacent VNF and shared-VNF.

Figure 4.9: Experimental Results - Comparison between sV-VNM Heuristic and FFA Heuristic.

(7-10) with 6-10% less utilized servers, as it shown in Fig. 4.9a. Similar results occurred regarding the physical links allocated for every request. On average, the sV-VNM utilized approximately 18% less physical links to allocate the virtual links of a VNE (Fig. 4.9b). The performance of the proposed sV-VNM is better in both aforementioned metrics, especially in the small and medium length VNEs, where the percentage of the shared-VNFs is higher, compared with the larger requests that the two heuristics perform similarly. Additional results concern the pairing ratio of adjacent VNFs that the heuristics achieve.

Figures 4.9c and 4.9d demonstrate the co-location ratio between adjacent VNFs in general and those that include shared-VNFs, respectively. Co-location ratio determines the efficiency of the VNFs mapping with regard to the servers' and links' utilization. Specifically, the co-location of adjacent VNFs leads to reduced network traffic between different servers and limited network delay between the corresponding VNFs, which is also beneficial regarding the QoS of a VN, as also stated in [23]. The proposed sV-VNM heuristic accomplishes an average co-location ratio of

69.8% in the general case, which is a noticeable improvement compared to the 62.7% of the FFAh. As for the pairs that involve shared-VNFs the corresponding ratio reached by the sV-VNM ranges from 95% for small length VNE requests to a minimum of 68% for larger VNE lengths. On the other hand, FFAh provides an average 60% co-location ratio of pairs that include shared-VNFs, which is approximately 20% less than sV-VNM. The results reflect the efficiency of the mixed policy adopted by the proposed algorithm. In particular, the partitioning based on the shared-VNFs and the placement of them in a Best-Fit fashion leads to a higher co-location ratio between the adjacent VNFs that connect the distinct VNE partitions. As mentioned above, each partition mapping starts from the host of the respective shared-VNF (if exists), while the worst-fit policy is used when the least recently used server stays on top of the candidate list for mapping the next VNF. This VNE partition mapping policy gives an advantage to the sV-VNM compared to the FFAh. Nevertheless, for a single VNE partition mapping, which does not include shared-VNF, the two heuristics are expected to perform equally.

4.6.2 kSP-DVNE Algorithm Evaluation

We hereby present the performance evaluation of kSP-DVNE algorithm. The proposed solution is compared with a technique introduced in [97], which tackles the issue of minimizing the network latency during hSFC embedding in an edge network environment. The authors in [97] design their solution on the Hybrid-SFC Embedding Auxiliary Graph (HSAG), the main components of which are defined as: (i) the *tier* that corresponds to the VNF (or Service Functions - SFs) of the hSFC, (ii) the *layer* of the substrate candidate EC nodes of every VNF (or SF) and (iii) the *nodes*, which reflect to the substrate EC nodes in our case. Each tier contains a set of layers, where each layer is composed of nodes, which correspond to the ECs that are candidates to host the related SF. A solution is constructed by examining all the possible paths that occur via the auxiliary graph. When the nodes of a layer are examined, for every one of them a new layer is defined in the next tier. Thus, starting from the initial tier, which includes only the source node, the paths to the candidate nodes for hosting the next VNF (or SF) are calculated. Then, for each tier's nodes the shortest paths to the next VNF in the sequence of the hSFC are calculated, until reaching the source node again. Consequently, we refer to the solution of [97] as HSAG, which achieves the minimum network latency in the hSFC embedding problem. Despite the fact that it is a faster solution than the brute force, it is still of high computational complexity in the worst case, in which the VNFs have a large number of candidate host nodes. This is due to the fact that the number of layers on each tier grows faster, while the same stands for the alternative paths, which are examined in the solution path construction. The discussed DVNE problem is equivalent with the hSFC problem with no-specific destination substrate node, as presented in [97]. We adapted the HSAG solution to our DVNE problem, and still it provides the solution with the minimal round-trip delay. However, our proposed algorithm achieves the optimal or near optimal solution much faster, making it suitable for IoT scenarios, not only for the initial embedding, but for different time sensitive network reconfiguration scenarios, where the VNE path has to be re-defined in the

substrate network. Furthermore, we implemented a greedy approach in addition to the proposed

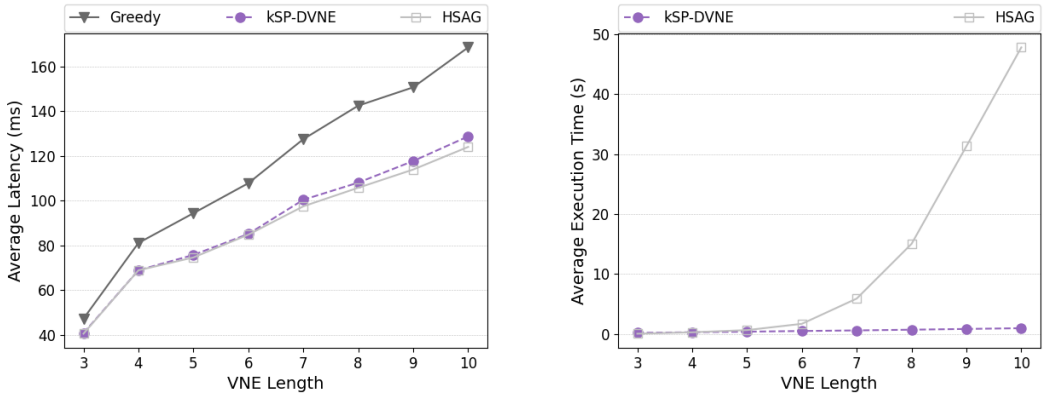
Table 4.4: kSP-DVNE Evaluation - Simulation Parameters.

Parameter	**Value**
Number of ECs (substrate nodes) $\|V\|$	50
VNE Length $\|V_s\|$ per request	$U[3, 10]$
Shared-VNFs in every request	$U[20\%, 30\%]$ of $\|V_s\|$
Physical Link network delay (ms)	$U[5, 15]$
Demanded cores per VNF	$U[1, 5]$
Total requests	700

kSP-DVNE and the HSAG. The greedy algorithm is also a path-based approach, while it utilizes Dijkstra algorithm to select in a greedy fashion, the next closest to the current substrate node, which is a candidate for the embedding of the examined VNF. In the augmented graph $\mathcal{G}_a$, starting from the source node v_0 the shortest path to the first VNF s_1 is calculated. The penultimate node of the path, is considered as the host u_1 of the s_1 VNF. Consequently, the shortest path from u_1 to s_2 is computed, and so on, until the construction of a complete embedding solution. The greedy algorithm is used on our kSP-DVNE approach, as a lower bound solution of round-trip delay minimization for the DVNE problem, to deal with some cases that might occur, where no valid candidate paths are identified in the augmented graph. Regarding the greedy algorithm's complexity, if m is the length of a VNE request, m Dijkstra shortest paths are calculated, so the time complexity is $\mathcal{O}(m(|V| + |E|)log|V|)$.

4.6.2.1 Simulation Setup

The alternative algorithms for solving the DVNE problem were implemented using the Python programming language. Relying on real network topologies [95], we simulated an edge network composed of 40 EC infrastructures. A request for VNE can be submitted to any of these ECs, which length fluctuates uniformly from 3 to 10 VNFs, with the 20% to 30% of these being shared-VNFs on specific ECs. These shared-VNFs derived form a pool of 20 in total. The computing resources demand for every VNF ranges uniformly for 1 to 5 CPU cores. Regarding the links between the ECs, the network delay varies from 5 to 15 ms. For each VNE length, 100 requests are generated. Based on this configuration, two different experiments are conducted. At first, we measure the efficiency of the proposed heuristic in comparison with the HSAG and the greedy solution, under a random mapping of the VNFs that compose a VNE request, following as much as possible a similar setting with [97]. Specifically, each EC can host between 2 and 3 out of the 20 available shared-VNFs. At the same time, an EC can be a candidate host of the 10% to 40% of the VNFs that compose a VNE request, following a uniform distribution. The average degree of the graph that represents the Edge Network equals to 3, while the kSP-DVNE algorithm is configured to operate with the value of k equals to $40 \times 3 = 120$, that is the number of the EC nodes $|\mathcal{V}_a|$ multiplied by the average degree of the graph $\deg(G)$. More details regarding the selection of the



(a) Average Round-Trip Delay.

(b) Average Execution Time.

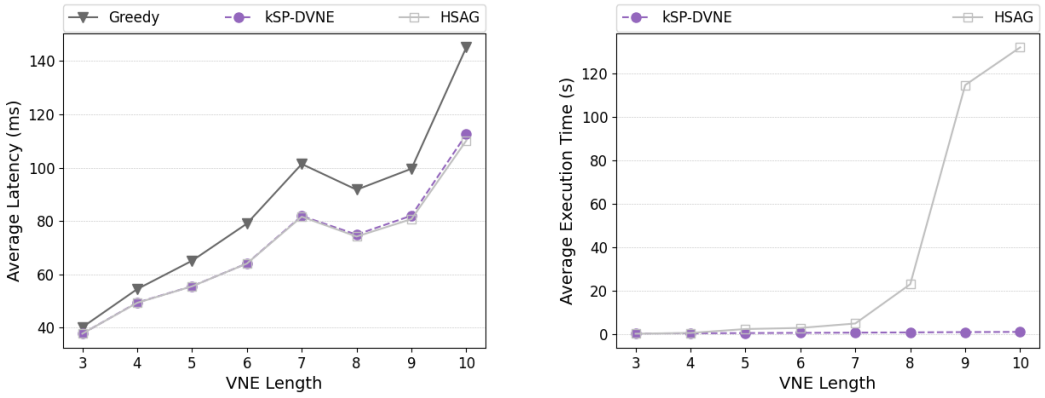
Figure 4.10: kSP-DVNE and HSAG comparison under random initial mapping of VNFs.

k value are provided in subsection 4.6.3. Table 4.4 includes all key simulation parameters for the evaluation of kSP-DVNE.

The second experiment aims at analysing the effectiveness of the proposed kSP-DVNE combined with the initial intra-EC heuristic. In detail, following the same experimental setup as above regarding the edge network parameters, as summarized in Table 4.4, the sV-VNM heuristic undertakes the initial mapping of the VN within the ECs. The intra-EC parameters are, also, identical to the intra-EC simulation parameters. In particular, these parameters are the *available cores per server* and the *available bandwidth per link*. During this experiment, the shared-VNFs availability is not constant among the ECs. Indicatively, per VNE request, the available shared-VNFs among the ECs are uniformly redistributed, in order to include a dynamic aspect in the simulations; that is to avoid the same shared VNFs to be always available in the same EC for all the VNE requests during the whole experiment. The number of shared-VNFs that each EC can provide for each VNE request varies from 2 to 3. Subsequently, the performance of the various compared DVNE approaches is analyzed, again under the mapping derived by the proposed sV-VNM heuristic.

4.6.2.2 kSP-DVNE Evaluation Results

Figure 4.10 shows the experimental results when the initial mapping of the VNFs s_i of a VNE request is performed randomly among the ECs. Both kSP-DVNE and HSAG outperform the greedy algorithm in terms of round-trip delay minimization. The proposed kSP-DNVE achieves the minimum round-trip delay at the 81% of the total requests. Compared to the HSAG, which provides the distributed embedding with the minimum round-trip delay, the kSP-DVNE produces embedding solutions with a slight increase of 1.88% in the average round-trip delay, for all the VNE lengths. Figure 4.10a depicts the round-trip delay of the embedding derived by each algorithm with respect to the VNE length. Concerning the execution time, which is illustrated in Fig. 4.10b, the proposed kSP-DVNE outperforms the HSAG by providing the embedding of the VNE much faster

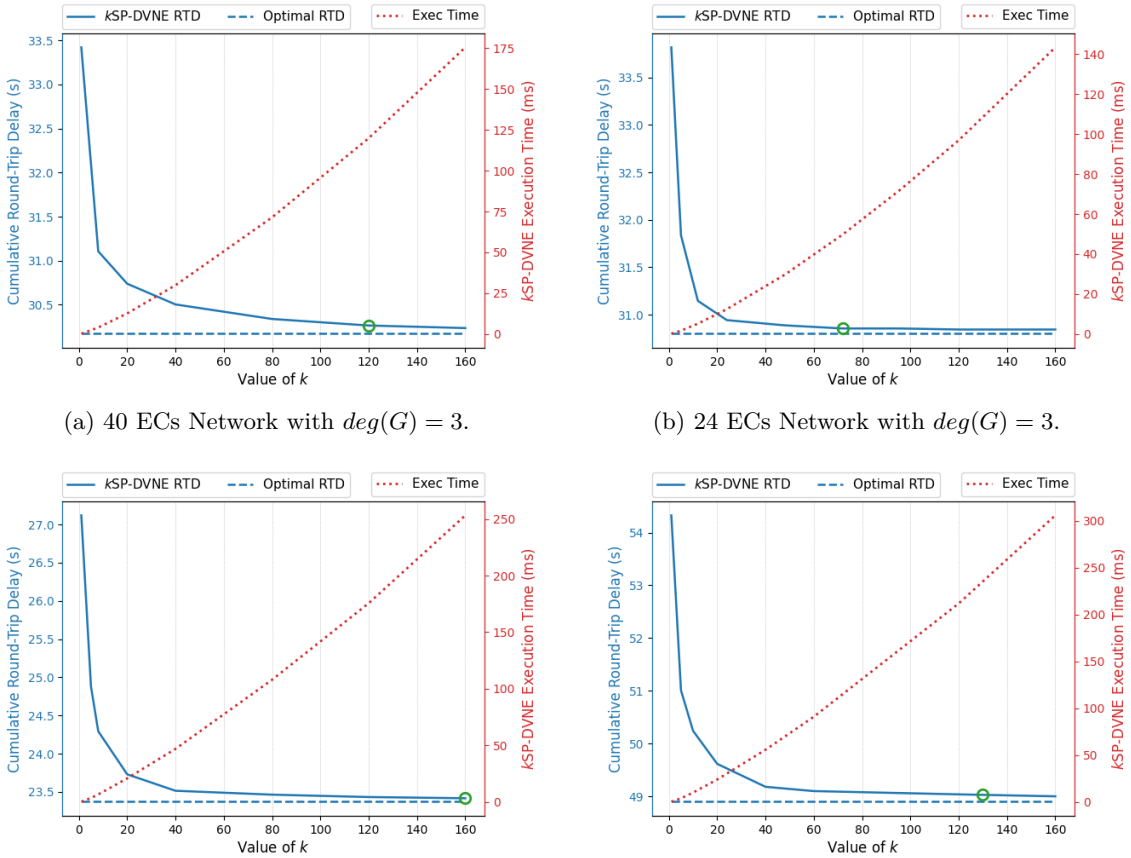


(a) Average Round-Trip Delay. (b) Average Execution Time.

Figure 4.11: Comparison of the kSP-DVNE versus HSAG under initial mapping using sV-VNM.

due to its lower complexity. Specifically, the execution time of the kSP-DVNE increases linearly with respect to the VNE length, as it is explained in the complexity analysis above, reaching an execution time reduction of over 90% on average compared to the HSAG, especially for a higher number of VNFs in the VN. It is worth mentioning that in the execution time of the kSP-DVNE heuristic, also the running time of the greedy algorithm is included since it is used to provide a lower bound of round-trip delay for our solution. Regarding the second experiment, where the initial mapping of the VNFs that compose a VN is performed using the proposed sV-VNM heuristic, the results are shown in Fig. 4.11. Concerning the comparison with the greedy algorithm, again, both kSP-DNVE and HSAG achieve better results in terms of the round-trip delay, as it shown in Fig. 4.11a. In fact, the kSP-DVNE performance shows significant improvement, as in the 95% of the cases provides the solution with the minimum round-trip delay, while the deviation from the HSAG solutions this time is approximately only 0.7%. Similar results with the first experiment occurred regarding the execution time, as illustrated in Fig. 4.11b, where, still, the kSP-DVNE outperforms the HSAG. The fact that the initial mapping comes from the sV-VNM, which reflects to a higher co-location of adjacent VNFs, enables the improvement of the kSP-DVNE performance in terms of the round-trip delay minimization. The increase of the execution time of the HSAG here, comes from the fact that the sV-VNM provides more candidate solutions compared to the previous experiment. Again, the execution time of the greedy algorithm is included in the corresponding kSP-DVNE heuristic.

In order to explain the reason of that improvement, firstly, we discuss about the proposed heuristic weaknesses. The main shortcoming of kSP-DVNE is that during the examination of the candidate paths for embedding, it always chooses the next embedding node towards the destination VNF, during the embedding of the f-VNP, or towards the source node v_0 for the embedding of the b-VNP, respectively. Thus, during the embedding of the f-VNP or b-VNP, if there is a network path of low network delay between two nodes that has to be traversed several times (back and



(a) 40 ECs Network with $deg(G) = 3$.

(b) 24 ECs Network with $deg(G) = 3$.

(c) 40 ECs Network with $deg(G) = 4$.

(d) 40 ECs 3-layer Hierarchical Network.

Figure 4.12: kSP-DVNE performance with varying values of k on several EC Networks.

forth) to provide the optimal solution, it will not be chosen by the kSP-DVNE heuristic, unless it is the only solution in the current candidate path. Therefore, by increasing the co-location of the adjacent VNFs of a VN, the need for multiple traversals of the same network links to provide the optimal results is eliminated.

Obviously, the major advantage of the proposed heuristic is the fast computation of the distributed embedding solution while achieving almost the same round-trip delay, on average, compared with the HSAG. This fact makes the kSP-DVNE a suitable solution to support ultra-reliable service deployment and high availability of NFV infrastructures at the network Edge, with minimized round-trip delay, especially in cases where re-configuration of the VN embedding is needed, such as end-device mobility, infrastructure failure and workload management in virtualized services. Especially, combined with the sV-VNM heuristic for the initial mapping of the VN, the kSP-DVNE provides most of the times the optimal solution in a real-time fashion.

4.6.3 Analysis for k parameter

The value of the parameter k determines the number of candidate paths that will be examined by the above DVNE algorithm, while it also determines its computational complexity. Obviously, a larger value of k corresponds to a larger search space for the algorithm and increased chances for finding a near optimal, or the optimal, DVNE solution, in terms of round-trip delay. Towards finding a suitable value of parameter k, we present a simulation-based analysis. Under 4 different edge network topologies, we conduct experiments with 500 different VNE requests, with the length of the VNs varying from 5-7 following a uniform distribution, while the rest of VNE and EC simulation parameters are identical to the above kSP-DVNE evaluation setting. Lets us also denote by $deg(G)$ the average vertex degree of the graph G, which represents the Edge Network, and by $|V| = n$, the number of the nodes of G, representing the ECs. The characteristics of the four topologies of the simulations are as follows: The first three topologies are single-layer networks with different densities, which correspond to various numbers of EC nodes and average node degrees. More specifically, their parameters are (i) $n = 40$, $deg(G) = 3$, (ii) $n = 24$, $deg(G) = 3$ (iii) $n = 40$, $deg(G) = 4$. Also, the specifications of these topologies are the same as in the first experiment of the kSP-DVNE evaluation algorithm (4.6.2.1). The last considered topology is a hierarchical EC network, composed of three layers, following the could continuum paradigm. The layers correspond to 1) the access, 2) the aggregation, and 3) the core layers. Each of these layers has different characteristics regarding the number of ECs, their computing capacities, and the network delay of the links between them. We denote each layer as a subgraph G_i that contains n_i ECs. For running simulations, the parameters are set as follows: *(1) access layer*: $n_1 = 25$, $deg(G_1) = 4$, *(2) aggregation layer*: $n_2 = 10$, $deg(G_2) = 3$, and *(3) core layer*: $n_3 = 5$, $deg(G_3) = 2$. The average network link delay between the EC nodes for each layer ranges in [5, 15] ms for the G_1, [15, 30] for G_2, and [30, 50] for G_3. The network delay of the links that connect the G_1 with G_2, and G_2 with G_3 fluctuates uniformly between 20 to 40 ms, and 40 to 60 ms, respectively. Moreover, to simulate the different capacities of ECs at each layer, we assumed that the shared-VNFs that an EC is able to host are in the range [2, 3], [3, 4], and [4, 5] for the respective layers represented by G_1, G_2, G_3. Correspondingly, a different number of VNFs for every VNE request is mapped, randomly, in the EC for each layer; that is [1, 2], [2, 3], and [2, 4] VNFs of a VNE request per respective layer's EC, following a uniform distribution.

In order to highlight the effect of k on the convergence and complexity of the proposed algorithm, the metrics that we consider are: (i) the cumulative round-trip delay of the kSP-DVNE heuristic, compared to the optimal solution, as well as (ii) its execution time, for varying values of k. Figure 4.12 shows the numerical results for the different edge network settings. At every plot, on the left y-axis the cumulative round-trip delay is shown, while the right y-axis, indicates the execution time of the kSP-DVNE heuristic, both with respect to the value of k. In all cases, we observe that the increase in the execution time of the algorithm is linear with respect to the value of k. The differences in the absolute execution times between the four simulations are due to the k-shortest paths algorithm's calculations, as a more dense graph with a higher number of nodes

demands more time to compute the k-shortest paths. Concerning the kSP-DVNE convergence, as shown in Fig. 4.12, the algorithm achieves near optimal solutions as $k \rightarrow (n \times deg(G))$, as it pointed with circle in the corresponding sub-figures. As described earlier, the proposed algorithm will not select a DVNE solution, where multiple traversals of the same edge have to be chosen, except the case that it is the only feasible solution in the examined path. An increased k value does not affect this behavior. Especially, in the case of the hierarchical EC network, we observe that the round-trip network delay is increased compared to the single-layer topologies, as more VNFs, shared and new deployed, have more candidate host ECs in the aggregation, or core layer, due to capacity constraints. However, based on the embedding policy of our algorithm, which avoids multiple traversals of the same link during the DVNE solution construction, its convergence to the optimal, exhibits similar behavior. Thus, in order to take advantage of the minimal execution time of the kSP-DVNE heuristic compared to HSAG and other exhaustive search and backtracking approaches, without discounts on its convergence, a value of k equal to $(n \times deg(G))$ is suggested. Under this setting and based on eq. (4.11), the computational complexity of the algorithm equals to $\mathcal{O}(deg(G)n^4)$.

4.7 Chapter Summary

In this Chapter, the problem of Distributed Virtual Network Embedding in a time-efficient manner is studied. In particular, concerning the strict network delay requirements of IoT-based applications, where services are provided as VNFs to the end-devices in an edge network, our work aims to provide efficient DVNE solutions with round-trip delay minimization. To achieve this, an algorithm that undertakes the initial intra-EC mapping of the VN is proposed, in order to identify the candidate host ECs for each VNF of the VNE request, while at the same time ensuring meeting the computing and network resource requirements. Furthermore, an algorithm for providing the DVNE solution is introduced, aiming at round-trip delay minimization. Given the initial mapping of the VN, an augmented graph is constructed to describe the properties of the Edge Network and the VNE request, and the k shortest paths algorithm is utilized to generate candidate embedding paths for the DVNE solution construction. Considering several m pairs of f-VNP and b-VNP, km number of candidate paths are examined. The proposed algorithm's worst-case complexity is $\mathcal{O}(kn^3)$.

Regarding the initial mapping algorithm, the comparative evaluation results indicate reduced server and network link utilization within the EC infrastructure, and a higher pair VNFs co-location ratio, compared to an alternative state-of-the-art approach, especially in cases where the VN contains pre-deployed and already mapped VNFs. With reference to the kSP-DVNE, the provided comparative evaluation highlights its efficiency, as it achieves to provide the optimal solution at the 95% of the VNE requests, especially when combined with the initial intra-EC mapping algorithm, with a significant reduction in the execution time compared to the HSAG solution. Moreover, a simulation-based analysis regarding the appropriate k value is presented.

Chapter 5

Enabling Industrial Network Slicing Orchestration: A Collaborative Edge Robotics Use Case

5.1 General Setting

Industry 4.0 is one of the most challenging 5G verticals that aspires to introduce modern network technologies in the manufacturing landscape. Industrial production processes usually rely on heterogeneous infrastructures of various vendors and have strict time and safety requirements. Modern Industry 4.0 applications require the interplay between industrial equipment, robots, and network infrastructure [109]. Cutting-edge network technologies, i.e., network slicing and edge computing, can facilitate the re-programmability of the industrial production workflows and provide the desired QoS level for robotic agents and manipulators, control loops, and actuators. Furthermore, current industrial operators usually do not possess the expertise required to handle such technologies, thus, industrial owners outsource the management of the industrial network infrastructure to third-party operators; This paves the way for micro operators to provide tailor-made network management solutions for smart factories [60].

In such complex environments, NFV and Edge Computing promise to provide the necessary processing capabilities at the network edge, in order to guarantee low network latency and enhanced data privacy while meeting the time-critical requirements of the dynamic manufacturing processes [14]. The deployment of virtualized industrial applications as VNFs allows the automated orchestration of end-to-end network services by standardized architectures, such as the ETSI NFV [19]. Furthermore, open-source operating systems and platforms, such as the Robotic

Operating System (ROS)[1] and Rapyuta [110], enable code reuse in robotics, collaboration with various robotic frameworks and the development of robotic applications on cloud infrastructures. To overcome the heterogeneity problems introduced by using multi-vendor equipment, the network softwarization of industrial applications can stimulate the creation of industrial-oriented network service marketplaces, where users can provide or consume any industrial application as a network service, independently of the underlying manufacturing infrastructure.

In this direction, this part of the thesis proposes an ETSI NFV-aligned architecture that enables the deployment of industrial-specific interconnected VNFs as network slices, at the edge of the network. Furthermore, it proposes a CSC mechanism that allows network slices of different owners to communicate towards the completion of collaborative tasks of a production process. Apart from providing secure communication between different vendors, CSC leads to the allocation of only the essential computing and network resources, which is important for the management of the limited resources of an industrial edge computing infrastructure. Our approach is evaluated through a use case in the area of Warehouse Robotics services [63], which requires the coordination of mobile robotic agents of different vendors on the factory floor. The results show that utilizing CSC achieves low-latency interaction while significantly reducing the completion time of a mission.

5.2 Related Work

The studies presented in this section are categorized based on two pillars: *i)* the orchestration of industrial applications at the network edge and *ii)* the establishment of intercommunication mechanisms of network services or network slices. Furthermore, we choose to focus on robotic applications because of their broad applicability and large impact in the smart manufacturing era.

In [111], the authors proposed a Cloud/Fog/Dew robotics model which leverages the infinite cloud resources for non-critical tasks, fog nodes for computing-intensive tasks that require low latency, while the time-critical tasks are performed at the lowest level -on the robots- towards increasing the reliability of the system regarding safety requirements. Adopting the IEEE OpenFog reference architecture, Shaik et al. [112] presented also a three-layer architecture for industrial robotics systems. They focused on the fog layer and described the essential services of system orchestration, application virtualization, memory management, real-time communication, scalability and dependability.

Taleb et al. [60] presented the key concepts of industrial networking which includes reconfigurable network slices for reprogrammable production processes, federation of heterogeneous edge slices over multi-domain factory environments, radio connectivity for industrial equipment, industrial equipment slicing and industrial network management without ownership. They proposed a multi-plane architecture for next-generation smart factories. From bottom to top, the Industrial Plane includes both industrial and networking equipment, while the Slice Plane refers to the network slices that host the industrial applications. The Orchestrator Plane is in charge of installing

[1] https://www.ros.org/

network slices over multiple domains. At the top level, the Industrial Business Plane provides requirements in the network slicing creation.

Antevski et al. [113] analyzed the ETSI NFV and ETSI MEC [10] architectures and they proposed a hybrid architecture that integrates the MEC reference architecture in an NFV environment. They illustrated the features of the proposed architecture, leveraging an Edge Robotics scenario that envisages a fleet of mobile robots remotely controlled and coordinated to perform two exemplary robotics MEC applications: *i)* a Robot Control application and *ii)* a Video Surveillance application in a multi-access indoor or outdoor environment.

Adopting the above "MECinNFV" architecture, the authors in [114] proposed a federated orchestration platform which allows service consumers of a specific administrative domain to instantiate their service to infrastructure providers of a different domain, through an auction-based mechanism that uses the Distributed Ledger Technology (DLT). Smart contracts were used to provide federation functionalities (i.e., registration, discovery, announcement, negotiation, acceptance, deployment, usage and charging). An edge robotic service was used as proof-of-concept and the proposed method was evaluated in terms of service deployment time over the federated domains.

Leveraging the automated functionalities of Open Source MANO (OSM) [82] and aligned with ETSI NFV and ETSI MEC reference architectures, the MESON platform provides centralised service registration and discovery, Edge Point of Presence (EPoP) selection and establishment of CSC within an EPoP [14]. On the other hand, the authors of [15] proposed a cross-service mechanism based on smart contracts that enables cross-service interactions and the establishment of a network service marketplace. Finally, FENDE [39] is a VNF-based marketplace that provides several functionalities that enable users to act either as providers or consumers and deploy customized service chains over cloud and edge infrastructures.

5.3 System Architecture

As mentioned above, the deployment of industrial applications as VNFs enables the exploitation of the virtualization technologies of the edge computing paradigm. Communication between different industrial network slices can improve coordination in an industrial environment and lead to more efficient resource provisioning. The proposed architecture towards the establishment of the CSC mechanism, is designed to meet a set of functional and non-functional requirements and incorporates three specific levels of orchestration and management, based on the ETSI NFV Standard [19]. Without loss of generality, we focus on robotic network slices for the rest of this Chapter. Fig. 5.1 depicts the proposed architecture overview, including all the necessary individual components of each layer. In short, the edge infrastructure layer is where the robotic applications will be deployed as network slices. The Virtual Infrastructure Manager (VIM) is the responsible component for managing the physical resources. The MANO layer is where an NFV Orchestrator Component, the OSM [82], interacts with the VIM in order to manage and control the NFV Infrastructure

(NFVI); here, the functionalities of each VNF are deployed by logically partitioning the hardware resources. Finally, on top of the MANO layer, the CSC orchestration layer contains all the essential components for the facilitation of efficient and secure interactions between the different slices.

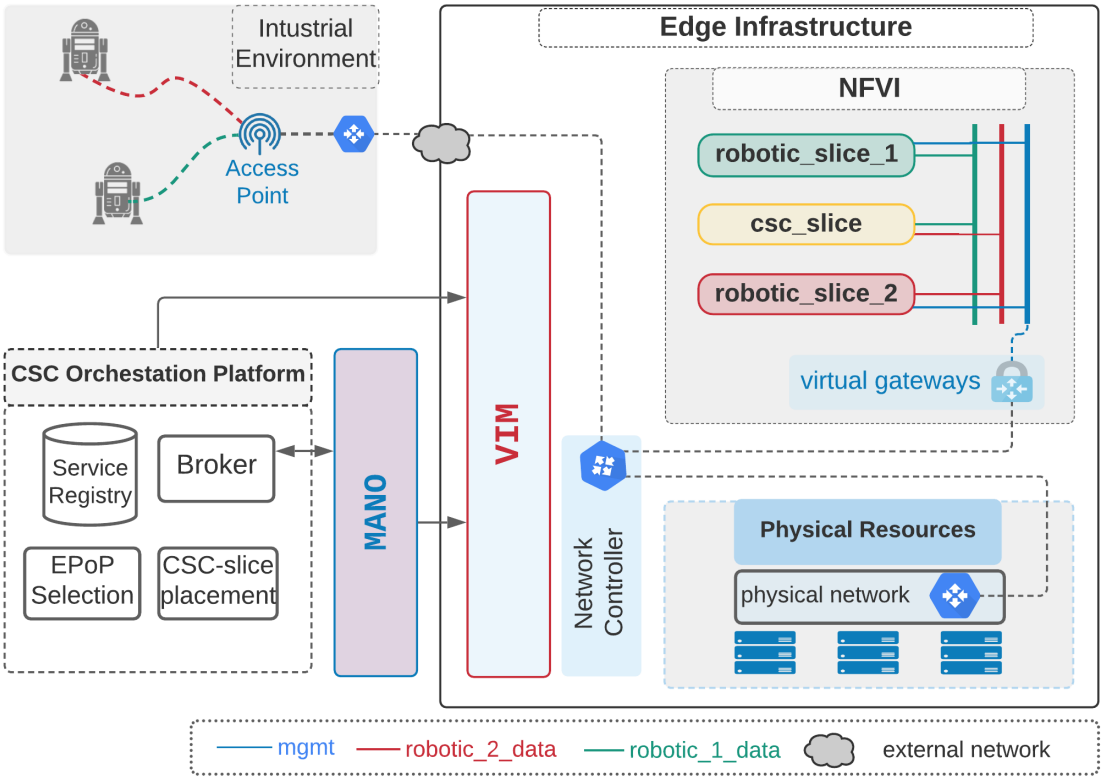


Figure 5.1: CSC-enabled architecture overview.

5.3.1 System Architectural Requirements

According to the above-described challenges for Industry 4.0, the virtualization and composition of industrial services as network slices require satisfying specific architectural design criteria for the slices' deployment and management, as well as for the establishment and orchestration of the CSC mechanism. In this context, the realization of CSC imposes certain functional requirements.

To begin with, a key requirement of such an architecture is to provide the capability of storage and discovery for the available CSC services of different vendors. The deployment of each slice takes place while trying to meet the resource requirements of the services (VNFs) it comprises of. Furthermore, to make the CSC mechanism efficient, the most suitable infrastructure for the placement of the slices must be selected; this selection should take into consideration the CSC requirements of the deployed slices. Thus, the establishment of CSC between slices requires the

extension of the basic OSM functionalities in terms of NFV management, by setting up a communication path between specific VNFs of different slices. These functionalities ensure the efficient and secure communication between the network slices, both at network and at application level, low response time, configuration capabilities for the inter-connected VNFs and optimal slice placement within the edge infrastructure for high quality service provisioning.

5.3.2 ETSI NFV Network Slicing

To enable network slicing over industrial domains, we introduce the main components of a Network Slice Instance (NSI) as defined in the OSM's information model [82]; these components are necessary for the appropriate configuration and management of CSC. The essential structural element of an NSI is the VNF, while the composition of multiple VNFs is defined as a Network Service. Essentially, MANO's network service specifies the virtual connection points and virtual links between multiple VNFs. The network slice is the top-level construct which encompasses a set of interconnected services. Each one of these virtualized elements is described by a descriptor file; a descriptor is actually a configuration template, that includes specific attributes about each virtualization unit.

In our approach, we focus on the attributes related to the CSC establishment. These include the virtual networks of each network slice and the specific virtual connection points of VNFs which are associated with a corresponding service (e.g., an API endpoint). Each slice incorporates isolated virtual networks to perform the hosted services' tasks (data networks) and is attached to a management network for external network connectivity. To achieve CSC between two slices, a CSC slice (CSC-slice) as described in [14] is defined. In essence, a CSC-slice is an intermediate network component for orchestrating the bidirectional network traffic between connection points of different slices. For example, as shown in Fig. 5.1, two slices, namely *robotic_slice_1* and *robotic_slice_2*, are deployed in a specific edge infrastructure. The CSC-slice is deployed to connect the two slices via connection points on the corresponding data networks (*robotic_1_data, robotic_2_data*). The instantiation of the CSC-slice is important to ensure secure and efficient communication among different slices and enable the automation and the management of the whole process.

5.3.3 CSC Orchestration

In this subsection, a brief description of the basic functionalities of the main components of the CSC Orchestration layer is given. Following the architecture of MESON [14], each individual component plays a significant role on the CSC mechanism establishment for industrial applications and aids in meeting a specific set of the above described requirements:

5.3.3.1 Service Registry

This component maintains the list of the CSC-enabled services from the different vendors. All the necessary data exposed by each vendor is contained in each record in the Registry. This data

comes in the form of a configuration template/descriptor and defines the connection points, virtual networks and specific VNF identifiers which host the corresponding industrial applications. The Service Registry, also, implements the binding of the CSC requests with service offerings from the respective vendors.

5.3.3.2 Edge Infrastructure Selection

A modern 5G industrial environment consists of multiple heterogeneous edge datacenters, which provide the resources for slice-deployment. This component identifies the most suitable edge infrastructure for realizing a slice deployment request from a vendor. It takes into account both the requirements of the vendor, including service co-location for CSC requests, and the infrastructure's data about the specific KPIs, such as minimum bandwidth, service availability, and elasticity. After a CSC request is made, multiple edge infrastructures are selected as candidates for the CSC deployment. These candidates are then evaluated based on the aforementioned KPIs, using a multi-criteria decision-making method [24].

5.3.3.3 CSC-slice Placement

This component undertakes the network slice placement within the edge infrastructure. Industrial applications usually pose strict latency constraints, therefore, an optimal placement within an edge infrastructure is of significant importance to further reduce the communication overhead imposed by the additional CSC functionalities (e.g., VNF management, firewall, and monitoring). After selecting the edge infrastructure, the CSC-slice Placement module interacts with the corresponding VIM. It retrieves data about the services to be linked through the CSC mechanism, regarding the specific edge servers (physical hosts) which host the services and then it performs the CSC-slice VNFs placement. Thereby, the CSC traffic will traverse the minimum physical path between the collaborating services through the CSC-slice instance, minimizing the additional time overhead of the CSC functionalities and reducing the overall latency, when compared to an arbitrary intra-infrastructure placement.

5.3.3.4 Centralized Broker

Lastly, this module is responsible for handling the necessary interactions between the CSC orchestration layer and the MANO layer, as well as to provide a client interface to the vendors for submitting slice and CSC deployment requests. The Broker is, also, responsible for the coordination of the interactions between the other components of the platform, depending on the deployment request type (e.g., a single slice deployment or a CSC request).

5.4 An Automated Storage and Retrieval System

To showcase a concrete example of a CSC application in Industry 4.0, we opted for a Warehouse Robotics application. In Warehouse Robotics, an Automated Storage and Retrieval System (AS/RS) is essential for enabling the smart factory vision. Efficiency in communication between the involved services is obviously of significant importance. Thus, AS/RS can greatly benefit from the delay minimization, increased fleet coordination, and data privacy and security that a CSC mechanism offers.

In detail, the discussed use case describes two warehouse robotic services and focuses on their interactions in a smart factory floor. For the sake of demonstration, two robotic services that usually cooperate in a smart factory setting are selected; the one undertakes the *storage* (loading) and the other the *retrieval* (unloading) of loads from defined storage locations. To better fit the application of our approach, we assume that these services are deployed and operated by different, individual vendors as network slices composed of specific VNFs in a chain.

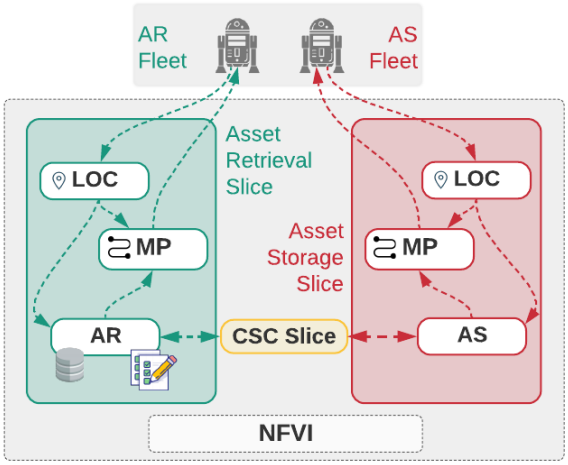


Figure 5.2: Slice/VNF interactions in the AS/RS use case.

In the envisioned scenario, either slice engages a mobile robot fleet that follows established routes in order to retrieve items or store them in specific locations. Whenever their respective mission is completed, the robots notify their corresponding slice which consequently/potentially triggers the CSC mechanism in order to inform the other slice about the remaining amounts of the assets in the inventory.

5.4.1 Asset Retrieval Slice

Specifically, the robots tied to the Asset Retrieval slice are responsible for unloading the assets from the shelves of the smart factory. The whole operation is dictated by the *Mission Planner (MP)* VNF which assigns missions to the robots to take specific amounts of from the inventory. The *Asset Retrieval (AR)* VNF undertakes the tasks of *i)* receiving the list of the required assets

to retrieve, from an external service, *ii)* maintaining an updated view of the inventory database and *iii)* invoking the CSC mechanism in order to inform the Asset Storage slice of potential asset shortages. Without loss of generality, we assume that the data exchanged through the CSC mechanism during the invocation can range from simple API calls to image or video files, depending on the implementation of the described VNFs. The *Localization Service (LOC)* VNF constantly computes the precise position of the robot and informs the MP and AR VNFs respectively [115, 116]. An overview of the components' interactions is depicted in Fig. 5.2.

5.4.2 Asset Storage Slice

On the other hand, the Asset Storage slice is responsible for loading the assets on the shelves. In this case, the MP VNF is responsible for dictating the trajectories that the robots will have to follow in order to refill specific assets in the inventory. The MP is invoked by the *Asset Storage (AS)* VNF which is triggered by the AR VNF of the Asset Retrieval slice when an item's availability is below a predefined threshold, through the CSC mechanism. When the storage mission is completed, the AS asynchronously informs the AR VNF of the specific item being available once again. The LOC VNF serves the same purpose as in the Asset Retrieval Service.

It is apparent that delay and floor space consumption minimization is critical for the above use case in order to efficiently coordinate the two tasks towards optimizing the mission completion time and subsequently operational costs. Apart from these, security is also of significant importance in the described scenario, as the data transferred between the two slices usually contain sensitive information regarding the state of the inventory, such as asset types and specifications or even images and videos, as previously mentioned. Thus, these reasons make this scenario a fitting candidate for showcasing the benefits of a CSC mechanism.

5.5 Experimental Evaluation

In this section, the experimental setup and the significant benefits of CSC in the context of AS/RS use case, are discussed in detail. To begin with, the proposed system architecture is deployed in the premises of the NETMODE testbed[2] of the National Technical University of Athens, Greece. The implementation of the aforementioned use case complies with the ETSI NFV standards. Thus, an OSM component operates as the NFV Orchestrator and communicates with an Openstack[3] deployment, operating as the VIM, which is responsible for instantiating the Virtual Deployment Units (VDUs) that contain all the necessary network and computing resources for the following experiments. As described in Section 5.3, these resources are defined by descriptors; specifically: *i)* the CSC requirements by the Network Slice Template (NST), *ii)* the network resources by the Network Service Descriptors (NSD) and *iii)* the computing resources by the Virtual Network Function Descriptors (VNFD). These descriptors are initially onboarded to OSM for the deployment of

[2]https://www.fed4fire.eu/testbeds/netmode/
[3]https://www.openstack.org/

the robotic services on the NFVI. Moreover, the communication between the services is performed via REST APIs deployed as Flask[4] applications. The robot fleet is composed of two identical AlphaBot[5] robotic platforms operating according to the missions of the AS/RS services respectively. The robots follow a circular path of points in the operating ground to perform the automated storage and retrieval system. Each point represents a storage location containing assets for retrieval. To highlight the importance and evaluate the proposed architecture design, we choose to conduct two different types of experiments.

5.5.1 Mission Completion Time Analysis

The first experiment highlights the effect of CSC on mission completion time. As described in Section 5.2, two different vendors are considered for the AR and AS slices. The operating floor consists of m different locations, each loaded with a unique type of assets. We assume that the mission of the AR slice agent is to collect a set of m different assets, one from each location. After collecting each set, it turns it over to the starting point for assembly. This results in the agent looping over each location on the floor and collecting one asset each time. We also assume that, in the meantime, assets can go out of stock and it is the AS slice's responsibility to reload them.

Depending on the levels of communication between the slices, two cases of scenarios are examined: *i)* the independent case and *ii)* the collaborative case. In the former, each slice operates autonomously on the factory floor as no communication takes place between them. That means that the AS agent has to periodically traverse and scan each location for asset shortages and then refill them. On the other hand, in the latter, the two slices leverage the CSC capabilities of the infrastructure; the AR slice triggers the AS slice when a shortage is detected after an asset retrieval and only then the AS agent is invoked to refill the respective location, as described in Section 5.4. It should be noted that, for these experiments, the time needed to travel between two adjacent locations is considered fixed. The time needed for the individual actions of asset storage and retrieval is also considered fixed.

In Fig. 5.3, the relation between the number of different storage locations and the average time needed for the completion of the AR mission, is depicted. The results have been averaged over 25 loops of AR missions, for each different number of locations. We observe a significant delay reduction in the collaborative CSC-enabled case compared to the independent one, which lies around 25% when 3 different locations are used and increases gradually to above 30% in the scenario where 50 storage locations are involved in the mission. These differences are mainly a consequence of the AR agent stalling its mission while waiting for the AS agent to detect the potential asset shortages in the independent case. Therefore, the significant role of the CSC mechanism in performance improvement and delay minimisation is highlighted.

[4]https://palletsprojects.com/p/flask/
[5]https://www.waveshare.com/wiki/AlphaBot2/

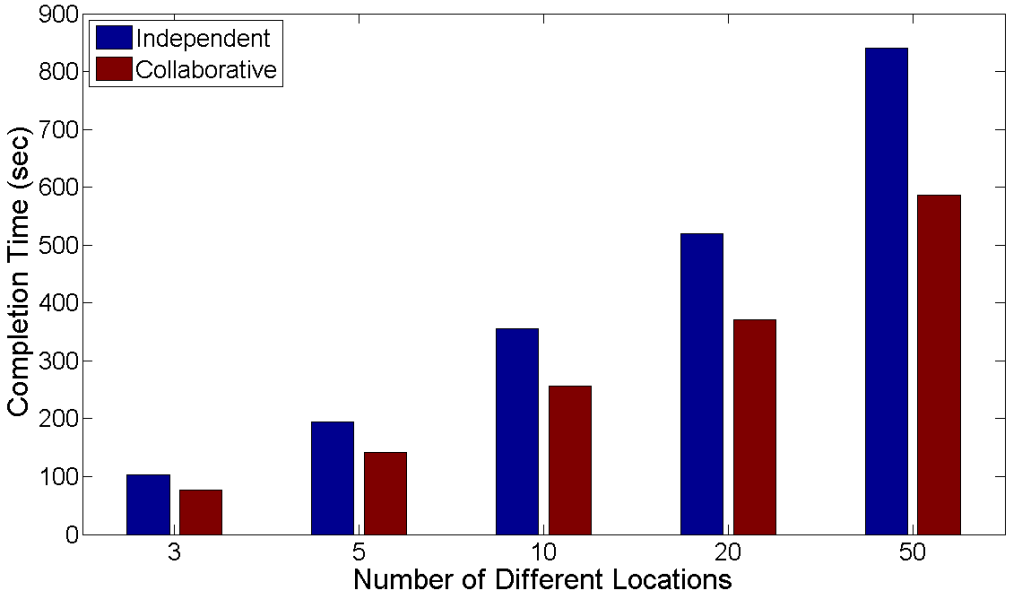


Figure 5.3: AR mission completion time sensitivity to number of locations.

5.5.2 Data Transmission Analysis

In the AS/RS use case, secure data exchange is equally important to the minimization of network delay. To this purpose, this experiment examines the effect of CSC in data transmission under different scenarios of CSC orchestration. In this setting, we assume that the AS and AR slices exchange information and data about the undergoing missions (collaborative case). The data size varies from 5MB (e.g., a small database) to 200MB (e.g., a video file) and, in order to produce unbiased results, we provide the averaged values from repeating the experiments 25 times. Fig. 5.4 demonstrates the transmission time under four alternative CSC settings.

As expected, when a direct link of communication (cyan bar) is established between the two services instantiated in the same edge server, the fastest transmission is achieved. However, this insecure CSC-slice placement does not offer isolation between the involved slices, thus, it is infeasible for the VIM to provide secure connectivity between the vendors, guarantee data protection and exploit the benefits of network slicing (e.g., monitoring and billing). What is more, the importance of an optimally placed CSC-slice (blue bar) is highlighted when the slices are deployed on the same edge infrastructure. We notice that, as the data volume increases, an arbitrary intra-infrastructure CSC-slice placement (yellow bar) provides inadequate results compared to an optimally placed CSC-slice, as the intermediate routers involved impose an extra overhead in transmission time and generate unreliable connections that produce fluctuation to the transmission rate. Finally, a multi-domain CSC-slice placement (red bar) between different edge infrastructures connected with optical fiber is evaluated. The results show that, especially for a small volume of exchanged data ($\leq$ 20MB) the optimally placed CSC-slice provides at least four times faster communication,

leading to mission optimization. This gap narrows when the volume of data exchanged increases, but data exchange is still close to 20% faster for large files (≈ 200MB). However, we believe that it is of great importance that the data exchange is performed internally in an edge infrastructure, consuming less bandwidth and network resources, compared to communicating through public internet. To sum up, the optimally placed CSC-slice provides similar results to the case of direct communication, while ensuring security, isolation, and reliability.

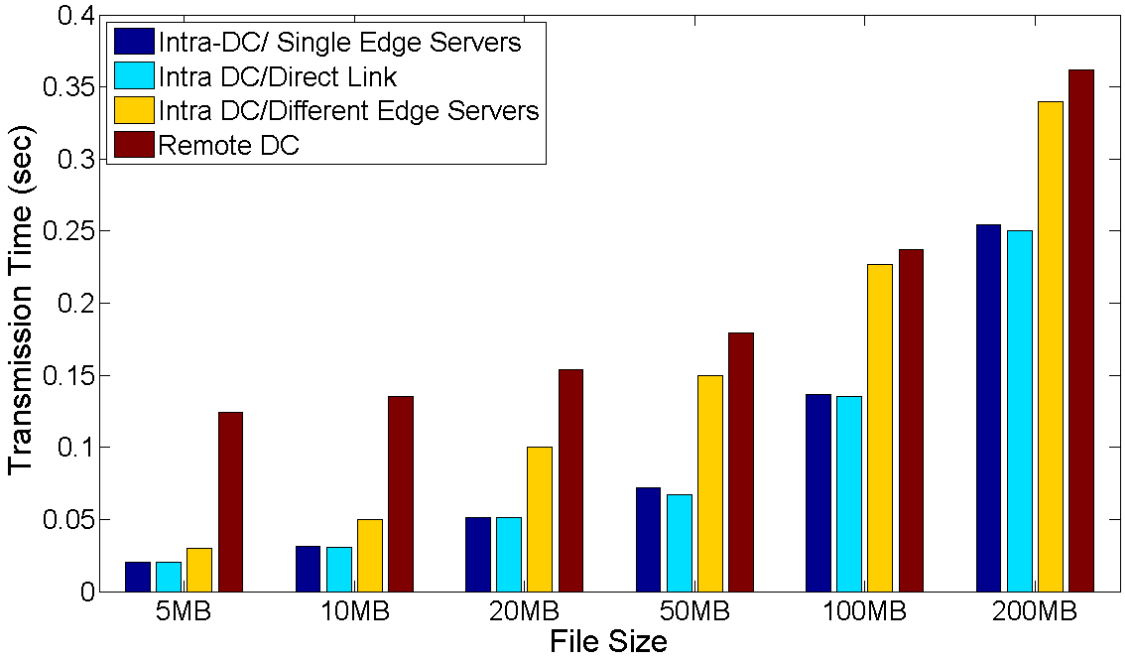


Figure 5.4: Transmission time sensitivity to different file sizes.

5.6 Chapter Summary

This part of the thesis presents an NFV-MANO-based architecture for enabling network slicing in edge robotics. Specifically, the proposed architecture aims to facilitate the deployment of robotic applications as network slices and to provide the necessary functionalities for CSC, enabling the collaboration between such applications in a smart manufacturing environment. In particular, the standard ETSI NFV reference architecture is extended in order to support communication capabilities between slices of different vendors, providing in this way the essential management and orchestration tools and ensuring secure CSC, based on each application's requirements. The proposed approach is evaluated on a use case in the context of Warehouse Robotics, where the collaboration of mobile robotic agents in an automated storage and retrieval system is essential. The results using the proposed CSC implementation showed significant improvements in the expected mission completion delay and data transmission duration when compared to other inter-communication

methods. Beyond the collaboration of robotic applications within defined industrial contexts, as exemplified in the aforementioned scenario, this architecture holds promise for various use cases. Among these are applications such as autonomous vehicles, which demand dynamic service provisioning in distributed edge/cloud infrastructures, facilitating seamless interaction with diverse robotic agents, mobile devices, and sensors.

Chapter 6

An AHP-based Autoscaling Framework for Kubernetes Clusters at the Network Edge

6.1 General Setting

As outlined in the introductory chapter the advent of 5G technology has ushered in a wave of IoT applications tailored for diverse human and business needs. While cloud computing, alone, provides ample computing resources, it falls short of meeting the low-latency demands of modern applications, edge computing offers finite computing resources located in close proximity to end-users, bolstering the processing capabilities of resource-constrained IoT devices [77].

So, effectively managing resources in the CC, for IoT-based applications poses a significant challenge. Many orchestration platforms, reliant on virtualization technology, offer comprehensive operations throughout the application's lifecycle. For instance, OpenStack serves as a widely-used Virtualized Infrastructure Manager for the management of virtual machines (VMs) within cloud infrastructure [44]. Additionally, Kubernetes is instrumental in orchestrating and managing containerized applications [45]. These platforms facilitate automated deployment, scaling, management, and maintenance of applications.

Focusing on autoscaling, both OpenStack and Kubernetes provide horizontal and vertical scaling. On both platforms, horizontal scaling is widely used. On the contrary, vertical scaling requires the restart of the VM (container), thus, it is not appropriate for real-time resource management. These scaling mechanisms are coarse and the scaling decision is usually based on simple monitoring metrics such as CPU and memory utilization without taking into account other important performance parameters and metrics, such as dynamic workload or energy consumption. Meeting the emerging 5G applications' rigorous requirements in terms of delay, throughput, and location

calls for the improvement of the existing scaling mechanisms to tackle the underlying challenges.

Towards dynamic scaling, many recent studies focused on extending the scaling capabilities of the existing platforms by including various performance criteria and application parameters. Regarding VM technology, a scaling mechanism for location-based applications was proposed in [117]. Based on various VM flavors, a scaling, and load-balancing mechanism determines the number of replicas of each VM flavor to serve the total incoming workload. The authors in [118] proposed two heuristics for VM horizontal scaling. The first heuristic focuses only on cloud computing resources to satisfy the resource requirements, while the second one also includes network resources and relies on multi-attribute decision-making (MAMD) algorithms to place new replicas to selected nodes. However, in this approach, the scaling decision is only based on resource availability without considering other system parameters or performance metrics.

Regarding Kubernetes, the Horizontal Pod Autoscaler (HPA) component is responsible for scaling the containerized applications. Phan et al. [119] proposed a traffic-aware HPA that adjusts the replicas of pods in edge nodes based on the proportion of incoming requests accessing nodes in real-time. The authors in [120] proposed an HPA-based scaling mechanism for edge clusters. A loss-less MMPP/M/c queuing model and an ML-based pro-active scaling method were proposed and compared with default HPA. Four forecasting methods were used to estimate the varying incoming request rate. However, the above studies do not take into account other performance metrics (e.g., power consumption) in the scaling decision.

Contrary to the default HPA and the above studies, this work aspires to include more criteria in the scaling decision focusing on power consumption and allocated resources. Towards this direction, this thesis introduces the AHP4HPA framework that is based on a custom autoscaling controller for Kubernetes and a multi-criteria decision-making approach, which considers various key performance indicators and application parameters. Specifically, the contributions of this work are summarized as follows:

- The scaling decision relies on resource profiles that provide a mapping between the QoS metrics and computing resources of an application pod to minimize any performance violations.

- The scaling decision of AHP4HPA is computed by Analytic Hierarchy Process [85], which is an MCDM method and considers various KPIs and parameters such as incoming workload, power consumption, number of active servers, and monetary costs. This light-computing approach can assess numerous scaling solutions in real-time and demonstrate the trade-off between application performance and cost minimization.

- AHP4HPA is evaluated in a small-scale edge cluster using a compute-intensive application and a dataset acquired from a touristic application. The results show that AHP4HPA significantly outperforms HPA in terms of power consumption and allocated resources, the cost of a slight increase in QoS violations.

6.2 System Architecture

In this work, the aim is the development of the AHP4HPA framework for Kubernetes clusters, a state-of-the-art system for resource management and orchestration of containerized applications. Hence, this section provides (i) an overview of Kubernetes, (ii) details regarding the AHP4HPA implementation in Kubernetes, and (iii) insights into the implementation of the introduced KPIs of the AHP algorithm.

In the context of Kubernetes, (i) pod is the smallest deployment unit of computing resources for a containerized application, (ii) service is an abstract way to declare pods that have the same set of functions (applications), and (iii) deployment is a declarative way for creating, modifying and scaling the pods. Pods can run multiple containers of the same application, which are managed as a single entity. In this work, it is considered that each pod runs only one container.

Definition 8. *Resource Profile φ_i is a mapping of the pod's allocated resources to the maximum request rate without violating the delay constraints.*

Let m denote the number of distinct resource profiles for an application. For all resource profiles φ_i, $i = 1, \ldots, m$, distinct services, and deployments are considered, while different resource limits for the pods are implemented. As stated in the Kubernetes documentation, [1] the resource limits are exploited to enforce that a running pod utilizes at most the predefined resources in terms of CPU and memory, which are defined as tuple $\langle c_i, r_i \rangle$ for CPU and memory respectively. Moreover, for each deployment, the number of identical pod replicas is defined as k_i and the upper replica limit is k_{max}. Thus, for each resource profile φ_i, $k_i \in [0, k_{max}]$, $k_i \in \mathbb{N}$. The resource profiles are created by experimenting with three different application settings, i.e., (i) the request rate, (ii) the resource limits, and (iii) the number of replicas. Through experiments, the maximum request rate before observing deterioration in the overall application performance is identified. To this extent, for each resource profile, a linear function $g_i(k_i) = \alpha_i k_i + \beta_i$ is computed by linear regression and maps the maximum request rate to the number of replicas. The App Deployment, which will be used by the AHP ranking below, is an abstraction that includes all deployments of all resource profiles, which serve the requests of a specific application.

Moreover, Kubernetes runs the containerized applications in Compute Nodes, which are hosted in OpenStack VMs in our case. To instantiate additional compute nodes and expand the Cluster, the Kubernetes Cluster Autoscaler (CA)[2] is employed. Cluster Autoscaler resizes the cluster, i.e., (i) adds a node to the cluster when the available resources are limited, (ii) removes a node, if the resources are underutilized. In this work, we select a compute-intensive application to showcase the performance of AHP4HPA, i.e., an image classification application developed with OpenCV[3]. Figure 6.1 presents the system architecture implemented in Kubernetes along with the workflow. The ingress traffic is redirected via a Load Balancer to the application pods, which are accessible

[1] https://kubernetes.io/docs/tasks/configure-pod-container/quality-service-pod/
[2] https://github.com/kubernetes/autoscaler
[3] https://docs.opencv.org/5.x/

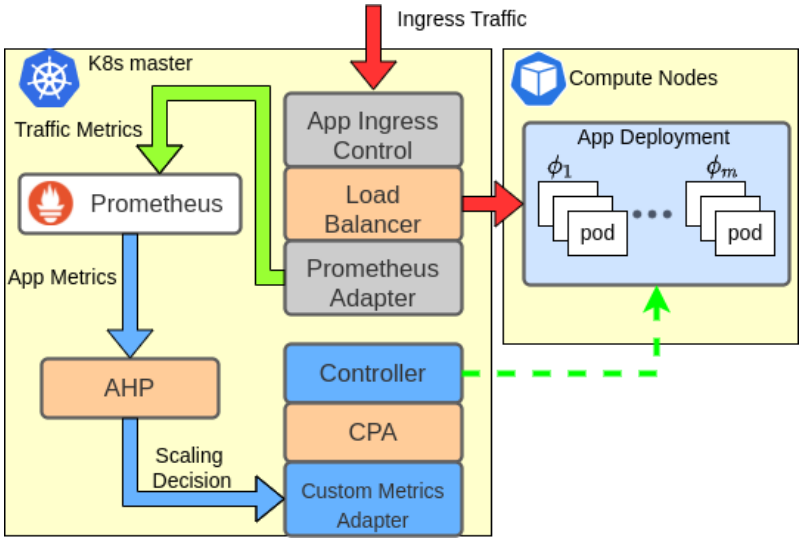


Figure 6.1: AHP4HPA integrated with Kubernetes.

externally from App Ingress Control. A Python Flask[4] REST API is deployed to act as the Load Balancer. Then, Flask is integrated with Celery[5], which creates asynchronous tasks assigned to workers. For each service, a Celery worker is responsible for redirecting the HTTP requests to the corresponding pod. Subsequently, relying on deployed replicas for every profile, dynamic balancing of the incoming workload on the instantiated pods is performed.

The proposed framework utilizes the Prometheus [121] monitoring system and Prometheus Adapter respectively for collecting traffic metrics and app metrics (per deployment, per pod). The core intelligence of the proposed framework is the AHP component. Periodically, the AHP algorithm considers the retrieved metrics and dictates the resource scaling decision for each deployment aiming at minimizing the total power consumption while meeting various QoS criteria. More information regarding the AHP-based scaling is provided in the following sections. The main scaling component in the Kubernetes ecosystem is the Horizontal Pod Autoscaler [122]. However, the HPA's algorithm is quite simplistic and mainly aims to stabilize the overall performance towards meeting a target value for a specific metric (e.g., CPU utilization). HPA scales horizontally (increases/decreases) the number of deployed replicas for a deployment. Therefore, the Custom Pod Autoscaler (CPA) [123] is employed to operate as the autoscaling controller, which enables custom algorithms for scaling to be realized. Moreover, CPA enables scaling to zero pods, which in the case of HPA is impossible and leads to higher power consumption. At each time slot, for each deployment, a CPA instance performs scaling according to the decisions of the AHP algorithm. All essential metrics for AHP assessment are exposed via the Prometheus Adapter and the scaling decision is consumed by the CPA via the Custom Metrics Adapter.

[4]https://flask.palletsprojects.com/en/2.1.x/
[5]https://docs.celeryq.dev/

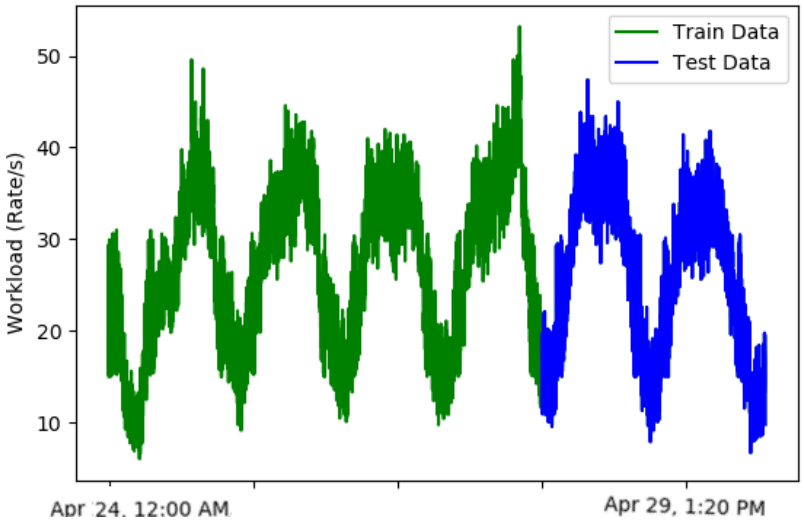


Figure 6.2: Workload Trace used for training and experimentation.

Finally, an ARIMA model [65], which is widely used for time-series forecasting, is implemented to provide a prediction for the request rate for the next time slot. To validate the presented framework, a workload trace from an application, which provides ferry booking services around Europe, is selected. The workload trace consists of HTTP requests produced by clients. As Kubernetes is mainly used for HTTP-based applications, this dataset is suitable for realistic evaluation. Figure 6.2 presents the distribution of request rate per minute over six days. The data from the first four days were used to train the ARIMA model, while the latter was for evaluation. At each time slot, the estimation of the ARIMA model is used by the AHP algorithms as described in 6.3.6.

6.3 AHP4HPA in Details

Towards efficient horizontal autoscaling both in terms of QoS and power consumption, we implement the Analytic Hierarchy Process, which is an MCDM method widely used for provider selection problems [24]. AHP facilitates the evaluation of alternative scaling decisions for a specific application, by considering several criteria of various types (e.g., numeric and boolean). These criteria are hierarchically structured and separated into *KPIs* and *Attributes*. In detail, KPIs indicate specific technical metrics, such as performance or cost-related metrics, while attributes summarize correlated KPIs. In the following, we define the essential KPIs and attributes related to our scaling problem. The scaling decision concerns the selection of the appropriate Application Deployment (App Deployment), which determines a set of replicas of different resource profiles for the corresponding application.

We consider m distinct resource profiles φ_i for an application. The different combinations of k_i replicas determine the total number of the candidate App Deployments. For $i = 1, \ldots, m$, $k_i \in [0,\ k_{max}]$, thus, the total number of the candidate App Deployments is $\mathcal{M} = (k_{max} + 1)^m$.

Table 6.1: AHP4HPA Parameters

Parameter	Definition
App Deployment Parameters	
D_j	An App Deployment j
φ_i	A resource profile i
k_{ji}	Number of replicas of φ_i in D_j
c_i	CPU Cores demanded for φ_i
r_i	RAM in GB demanded for φ_i
$sc_j(\varphi_i)$	Transformation cost for φ_i of D_j
TC_{D_j}	Total transformation cost of D_j
a	Transformation Cost factor, $a \in (0.5,\ 1)$
Power Consumption Parameters	
S_{D_j}	Number of active servers required by D_j
P_{MAX}	Maximum Power Consumption of a server
P_{D_j}	Power Consumption of the D_j
$P_{D_j}^{total}$	Total power consumption under D_j
QoS & Billing Parameters	
$b(\varphi_i)$	Billing per instantiation of φ_i
B_{D_j}	Total billing for D_j
$f(D_j)$	Profiling of service rate for D_j
QoS_{D_j}	QoS score of D_j
Server Parameters	
C_{total}	Available CPU cores in a server
R_{total}	Available RAM in a server

An App Deployment D_j, where $j = 1,\ \ldots,\ \mathcal{M}$, is defined as:

$$D_j = \langle k_{ji} \rangle,\ i = 1, \ldots, m\ .$$

Given the set of candidate App Deployments U_D, the suggested framework strives to nominate the App Deployment $D^* \in U_D$, which minimizes the power consumption and the allocated computing resources from the infrastructure provider's perspective and at the same time, guarantees a certain QoS level for the user. The quantification of the above evaluation criteria is done by the definition of the appropriate KPIs and the composition of the hierarchical structure, as they are presented below. Table 6.1 summarizes the parameters of the AHP4HPA algorithm.

6.3.1 Total Allocated Resources

In the context of micro-services, a resource profile reflects the allocated resources of containers. The most common resources to specify are CPU and memory. As we mentioned above, an App Deployment D_j is a set of replicas of different resource profiles φ_i. Each φ_i has a specific resource request c_i of CPU cores and r_i of memory. So, we define two KPIs, to express the total resource demands for a D_j; first, the *CPU Cores* C_j and secondly the *RAM* R_j, where:

$$C_{D_j} = \sum_{i=1}^{m} k_{ji} c_i \quad (6.1) \quad \text{and} \quad R_{D_j} = \sum_{i=1}^{m} k_{ji} r_i \quad (6.2)$$

6.3.2 Active Servers

From the provider's perspective, the minimization of power consumption is directly related to the reduction of active servers. The number of active servers demanded for an App Deployment is derived from the total resources required in relation to those provided in a single server of the infrastructure. Let C_{total} be the total CPU Cores and R_{total} the total GB of RAM that are available in a server. Based on the resource demands for a specific D_j, the required number of active servers S_{D_j} is:

$$S_{D_j} = \max\left(\left\lceil \frac{C_{D_j}}{C_{total}} \right\rceil, \left\lceil \frac{R_{D_j}}{R_{total}} \right\rceil\right). \quad (6.3)$$

The *Active Servers* KPI comprises a major power-related metric, as almost 70% of the maximum power consumption of a server occurs on its idle state [124]; a new server activation brings increased power demands for the infrastructure.

6.3.3 App Deployment's Power Consumption

Several energy-aware approaches for cloud computing propose a power model based on the server's maximum power consumption when it is fully loaded (P_{MAX}) and its idle state consumption (P_{min}). The following model is introduced in [124] to predict the server's power consumption:

$$P = \gamma * P_{MAX} + (1 - \gamma) * P_{MAX} * u. \quad (6.4)$$

The γ parameter refers to the proportion of the consumed power of an idle server with respect to P_{MAX}. The second term of equation (6.4), denotes the server's power consumption which occurs from its CPU utilization u. In accordance with [124], u depends on the allocated resources of a deployed application, with respect to the available resources of the server. Hence, in our model, we define the power consumption for an App Deployment D_j based on its resource requirements

and the CPU utilization of the corresponding active servers:

$$P_{D_j} = (1-\gamma) * P_{MAX} * \left(\frac{C_{D_j}}{S_{D_j} * C_{total}}\right). \tag{6.5}$$

Therefore, based on (6.4), (6.5), the total power consumption under the specified deployment is defined as,

$$P_{D_j}^{total} = S_{D_j} * \gamma * P_{MAX} + P_{D_j}. \tag{6.6}$$

6.3.4 Transformation Cost

The scaling process itself brings about extra resource management overhead, as different workloads require deployment adjustment. To quantify these adjustments required per case, we define the Transformation Cost KPI. It reflects on a penalization of the adjustments of each candidate App Deployment, from the current state. Let the scaling cost of a resource profile φ_i from current App Deployment D_c to the D_j is:

$$sc_j(\varphi_i) = \begin{cases} 1, & if\ k_{ci} = k_{ji} \text{ (no scaling)} \\ 1 + (k_{ji} - k_{ci})a, & if\ k_{ci} < k_{ji} \text{ (scale-out)} \\ 1 + (k_{ci} - k_{ji})(1-a), & if\ k_{ci} > k_{ji} \text{ (scale-in)} \end{cases}. \tag{6.7}$$

where k_{ci} is the number of replicas of φ_i in the current App Deployment D_c and k_{ji} the replicas of φ_i for the candidate App Deployment D_j. The parameter a is a degree of penalization regarding the scaling-out adjustments with respect to scaling-in changes. If $a = 0.5$, the penalty is equal for both cases. So, the Transformation Cost for the candidate D_j is:

$$TC_{D_j} = \sum_{i=1}^{m} sc_j(\varphi_i). \tag{6.8}$$

6.3.5 App Deployment's Billing

Typically, cloud providers charge VM or container's instance based on the instance type (e.g., OS type) as well as the required resources (in the form of resource profiles) for a specific period of time. Therefore, for each resource profile φ_i, the corresponding billing of a replica of φ_i is defined as $b(\varphi_i)$. Thus, for D_j the total billing for a user is:

$$B_{D_j} = \sum_{i=1}^{m} k_{ji} b(\varphi_i)\ . \tag{6.9}$$

6.3.6 Resource Profiling - QoS

Towards the selection of the appropriate scaling decision, it is important to include a QoS-related metric for the evaluation of the candidate App Deployments. To address this, we rely on (i) a

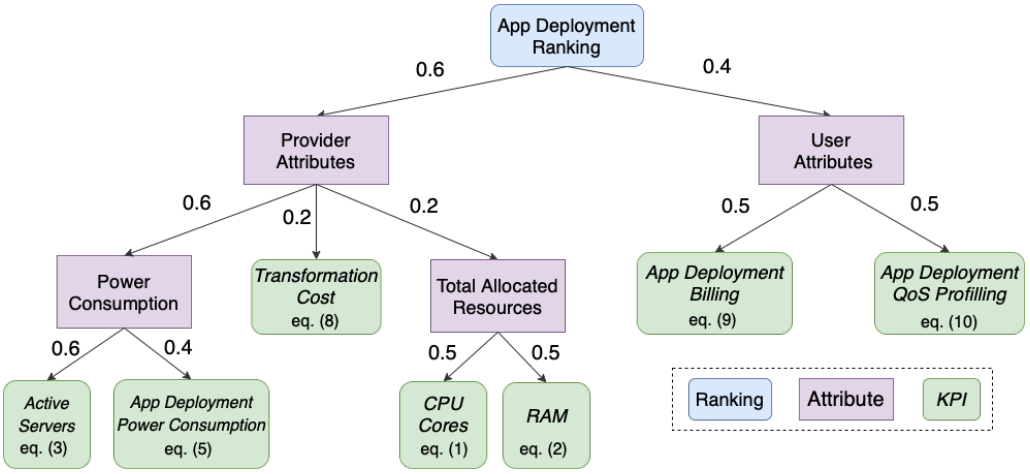


Figure 6.3: AHP Structure Model.

profiling regarding the potential service rate for every available resource profiles φ_i and (ii) on a prediction model of the incoming workload $\tilde{\lambda}$ for the respective application in a specific time-slot. The predicted workload rate is expressed as requests per second. Concerning the resource profiling, which corresponds to the total service rate for the next time slot for each candidate App Deployment, we calculate the QoS profile for each one of them as: $f(D_j) = \sum_{i=1}^{m} g(k_{ji})$. So, in order to evaluate the candidate App Deployments, the QoS score KPI is determined as,

$$QoS_{D_j} = \begin{cases} 0.2, & \text{if } f(D_j) < \tilde{\lambda} \\ 1 + \frac{0.5}{1+(f(D_j)-\tilde{\lambda})}, & \text{if } f(D_j) \geq \tilde{\lambda} \end{cases}. \tag{6.10}$$

This KPI penalizes candidates that do not guarantee the QoS for the predicted workload. Regarding the case where $f(D_j) \geq \tilde{\lambda}$, the score is lower for over-provisioned resources for the respective $\tilde{\lambda}$. Specifications about the profiling and the workload predictor model can be found in section 6.2.

6.4 AHP-Based Autoscaling

In this section, an overview of the MCDM method of the proposed framework is presented. Specifically, the ranking process of the candidate App Deployments for scaling is based on AHP. The AHP method enables the simultaneous processing of various criteria, by structuring them in a hierarchical manner. In fact, conflicting criteria groups can be considered for the final decision. Thus, provider-related attributes such as Power Consumption, and Resource Allocation can be combined with user-related requirements for high QoS with minimum billing. In the following, we present the AHP adaptations for AHP4HPA. The AHP calculations of the proposed solution can be divided into three main phases:

Phase 1 - Hierarchical Structure & Weight Assignment: Based on the above-described

KPIs and attributes, the hierarchical structure of the proposed framework is depicted in Figure 6.3. The leaves on the hierarchy tree represent the KPIs of the model, which are condensed into an attribute of a higher level. For example, the *Active Servers* and App Deployment's *Power Consumption* KPIs determine the *Power Consumption* attribute. Furthermore, AHP enables the configuration of the importance of each node, by weight assignment on the edge between two criteria. Specifically, assuming that the weight of the attribute or KPI A at level i is, $w_{iA} \leq 1$, then the sum of weights for a set of siblings KPIs and attributes of an attribute A at level i is, $\sum w_{iA} = 1$. Figure 6.3 also includes the weight assignment for our framework. The main scope of AHP4HPA is to minimize the total power consumption while taking into consideration multiple other parameters that are reflected in the proposed weight assignment.

Phase 2 - Relative Attribute Score Computation: Candidate App Deployments are assigned a value for each KPI of the structure based on equations (6.1) - (6.10). A Relative Comparison Matrix is computed for every KPI, to represent the pairwise comparison between the competing App Deployments that take place in the process. In this work, the KPIs are numeric values of two types: (i) *Higher is better* and (ii) *Lower is better*. In our model, all KPIs are numeric-*Lower is better* criteria, except of the *App Deployment QoS Profiling*, which is of numeric-*Higher is better* type. Let A_j denote the value assigned in KPI X for the candidate D_j. Then, for $j = 1, \ldots, \mathcal{M}$, where $\mathcal{M} = m^{k+1}$, we compute the RCM of KPI X -if X is of *higher-is-better* type- following the definitions of Chapter 2, specifically provided in Table 2.1 and eq. (2.1):

$$RCM_X = \begin{bmatrix} 1 & A_1/A_2 & \ldots & A_1/A_{\mathcal{M}} \\ A_2/A_1 & 1 & \ldots & A_2/A_{\mathcal{M}} \\ \vdots & \vdots & & \vdots \\ A_{\mathcal{M}}/A_1 & A_{\mathcal{M}}/A_2 & \ldots & 1 \end{bmatrix}. \tag{6.11}$$

For *Lower is better* KPIs, the RCM occurs by the computation of the transposed matrix of (6.11). Furthermore, given the $RCM_X = [x_{ij}],\ i, j = 1, \ldots, \mathcal{M}$, a Relative Ranking Vector for each KPI is computed based on the mean of normalized values method of (2.2):

$$RRV_X = [v_{X1} \ldots v_{X2}], \text{ where } v_{Xi} = \frac{\sum_{j=1}^{\mathcal{M}} x_{ij}}{\sum_{i=1}^{\mathcal{M}} \sum_{j=1}^{\mathcal{M}} x_{ij}}. \tag{6.12}$$

More details regarding the AHP calculations of different types of KPIs can be found in Chapter 2.

Phase 3 - App Deployments Ranking and Decision: For all Attributes, the RRV is computed in a bottom-up fashion, until the computation of the RRV of the App Deployments Ranking (top-level attribute). The computations are based on the lower-level RRVs and the weights among siblings KPIs and attributes. For a parent attribute with n sub-attributes and the weight vector,

which consists of n values, the RRV is computed as:

$$RRV_{par} = \begin{bmatrix} v^1_{sub1} & \dots & v^1_{subn} \\ v^2_{sub1} & \dots & v^2_{subn} \\ \vdots & & \vdots \\ v^n_{sub1} & \dots & v^n_{subn} \end{bmatrix} \begin{bmatrix} w_{sub1} \\ \vdots \\ w_{subn} \end{bmatrix} = \begin{bmatrix} v^1_{par} \\ \vdots \\ v^n_{par} \end{bmatrix}. \tag{6.13}$$

Finally, an RRV is computed for the ranking of the candidate App Deployments for scaling. This RRV has the form of: $[r_1,\ r_2, \dots,\ r_{\mathcal{M}}]$. The maximum $r^* = \max[r_j]_{j=1}^{\mathcal{M}}$ value corresponds to the nominated App Deployment D^*.

6.5 Experimental Evaluation

In this section, the experimental setup and comparative performance evaluation of the AHP4HPA framework is presented. The experimentation is conducted on the premises of the NETMODE[6] testbed of the National Technical University of Athens, Greece. The Kubernetes Cluster is deployed in self-hosted OpenStack VMs in two Intel Xeon Silver 4110 servers to enable the Kubernetes Cluster Autoscaler. We consider three types of resource profiles, $m = 3$, i.e., small, medium, and large, and the maximum number of replicas for each φ_i is $k_{max} = 4$. The resource limits and billing for each resource profile along with the capacity of the Kubernetes Nodes, are presented in Table 6.2. For the billing of the resources, we take information from Azure[7] pricing calculator, and we consider that solely the instantiation of a new replica is charged for the whole amount. The maximum power consumption for a fully operational server is $P_{MAX} = 2000W$ in accordance with [125]. For training the workload prediction model with the above dataset, autoarima[8] package is employed to provide the framework with an ARIMA model with order=(3,1,1). This model provides sufficient results. Moreover, it is assumed that when the response time of a request for the image classification application takes three times more than expected, the request is considered lost, and a connection timeout is enforced.

Table 6.2: Setting for resource flavors

Resource Profiles	**Small**	**Medium**	**Large**	**Kubernetes Node**
CPU cores	1	2	4	8
RAM (GB)	2	4	8	16
Billing $ per instantiation	7	13	21	

Figure 6.4 presents the maximum request rate for the respective number of replicas for each

[6] https://www.fed4fire.eu/testbeds/netmode/
[7] https://azure.microsoft.com/en-us/pricing/calculator/
[8] https://pypi.org/project/pmdarima/

resource profile. A linear regression technique is used to extract the respective QoS mapping $g_i(k_i)$ for this application for the three resource profiles. QoS mapping functions are also utilized for dynamic load balancing of the ingress traffic, according to the deployed number of replicas for each App Deployment.

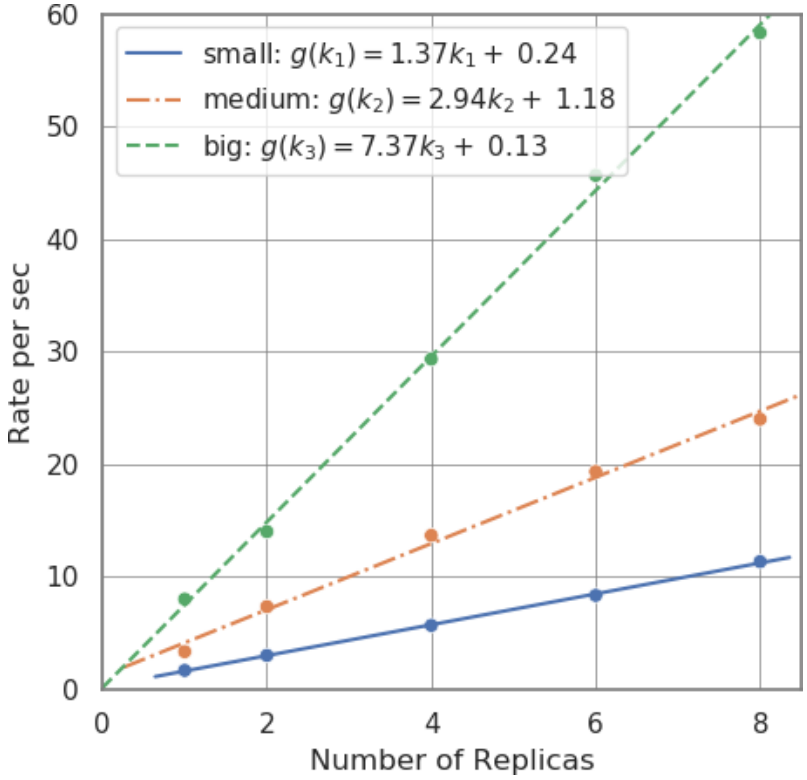


Figure 6.4: The maximum rate per sec for the selected application to the number of replicas for each resource profile.

Three different setups are compared, namely: (i) the proposed AHP4HPA, (ii) modified HPA (M-HPA), and (iii) the default HPA (D-HPA) of Kubernetes. For M-HPA, the resource profiles are also included along with the dynamic load balancing. The HPA is trying to target 70% CPU utilization for the deployed pods. On the other hand, for D-HPA none of the proposed techniques is employed, so we select to target 70% CPU utilization, utilizing only the medium resource profile. To produce unbiased results, all three scaling mechanisms operate every 15s (time slot), while we evaluate them against a three-hour workload traffic dataset from the trace test dataset. Also, the scaling stabilization window is disabled for the three setups. As the deployed application is mainly compute-intensive, in the following results, the focus is on CPU utilization. All three scaling decisions run in the order of ms, as a result, the overall performance is not disrupted.

In Figure 6.5, the power consumption of the deployed resources for the three experiments is presented. We calculated the power consumption using (6.6) according to the deployment instantiated

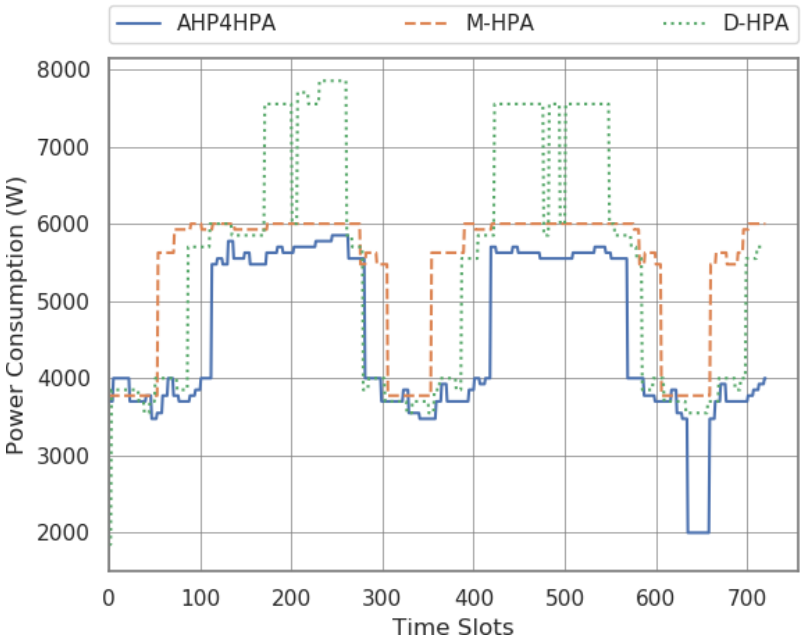


Figure 6.5: Power Consumption in W for each experiment at each time slot

by each autoscaling method at each time slot. The results for each experiment are summarized in Table 6.3. Specifically, the average energy consumption throughout the experiment results in 3.96 kWh for AHP4HPA. Similarly, in Figure 6.6, the total number of cores instantiated at each time slot for each setup is presented. For AHP4HPA, the CPU cores deployed for all resource flavors are 15.8 vCPUs averaged throughout the experiment. Our solution outperforms both M-HPA and D-HPA, producing 9% and 14% less average energy consumption accordingly. The proposed framework sufficiently serves the incoming workload having only 1.53% percent of total lost requests, while M-HPA has 0.37% and D-HPA has 0.25%. The difference in total request loss is mainly due to the workload prediction miscalculations produced by the ARIMA. We should also mention that the average energy consumption of M-HPA is 4.32 kWh with average CPU cores of 20.5 vCPUs while the D-HPA is 4.57 kWh and 19.4 vCPUs. Hence, these results indicate that the inclusion of resource profiles leads to 6.5% less average energy consumption, regardless of the increased utilization of CPU resources. This is explained since D-HPA autoscaling results in instantiate a 4^{th} node in the Kubernetes Cluster, consuming unnecessary power to serve the incoming traffic.

Table 6.3: Results for the three experiments

Setup	**Total Request Loss**	**Average Energy Consumption**	**Average CPU cores**
AHP4HPA	1.53%	3.96 kWh	15.8
M-HPA	0.37%	4.32 kWh	20.5
D-HPA	0.25%	4.57 kWh	19.4

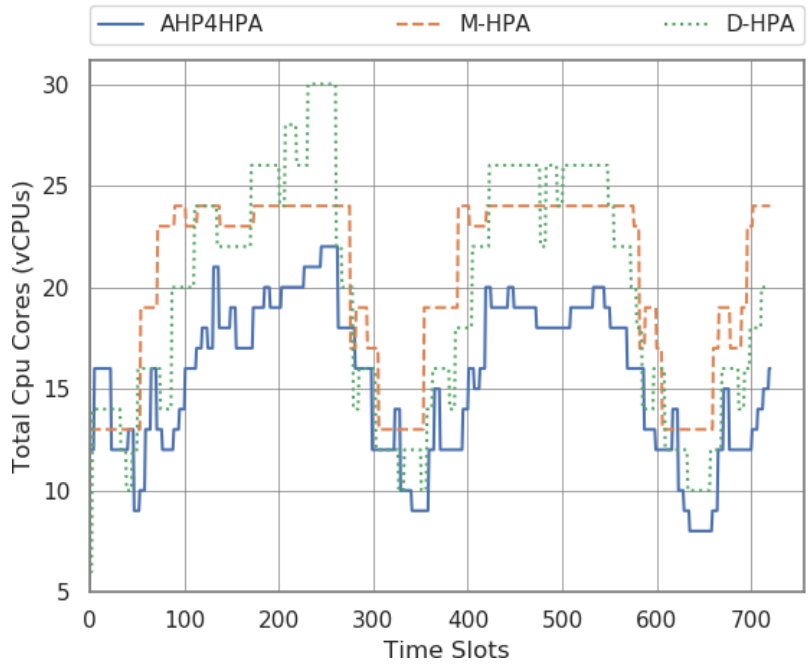


Figure 6.6: Number of cores instantiated for each experiment at each time slot

6.6 A Multi-Application Autoscaling Framework

The AHP4HPA framework that was described earlier serves as the foundation for the implementation of dynamic resource scaling mechanisms. This sophisticated architecture seamlessly integrates a hybrid approach that blends reactive and proactive methods, yielding substantial enhancements across key performance metrics. These improvements manifest notably in areas such as energy consumption mitigation, resource optimization, and the assurance of QoS for individual applications.

However, it's imperative to recognize that in the contemporary landscape of the cloud continuum, the challenges extend beyond the scope of single-application solutions. The demand for holistic resource orchestration and management solutions has surged, primarily driven by the proliferation of diverse and interrelated applications. This shift in perspective underscores the need for architectures capable of orchestrating scaling decisions across multiple applications that are developed in the same edge infrastructure, concurrently. Such an architectural evolution is pivotal, as it addresses the intricate interplay between various applications, each with its distinct resource demands and QoS criteria.

In essence, this architectural progression represents not a mere incremental refinement but rather a strategic response to the evolving intricacies of modern cloud environments. Its objective is to transcend the constraints of single-application solutions by facilitating the seamless coexistence and efficient allocation of resources for numerous applications within the cloud continuum.

This shift towards orchestrating multiple applications concurrently holds particular importance in optimizing infrastructure resources. The interactions among these applications can lead to resource contention and inefficiencies, underscoring the significance of adopting a holistic approach.

Thus, in this section, a preliminary extension of the AHP4HPA framework is introduced to support scaling decisions for multiple applications that are deployed within a Kubernetes Edge Cluster (KEC), in order to examine potential benefits regarding further optimization in terms of the infrastructure's energy consumption and resource utilization.

6.6.1 Multi-Application Autoscaling Architecture

In Fig. 6.7 an overview of the multi-app framework architecture, which is introduced in this work, is presented. In the setting under consideration, the ingress traffic arrives in a KEC, where several applications are placed in Kubernetes compute nodes. An Ingress Controller is responsible for redirecting (and load balancing) the incoming workload to the application pods using the respective Kubernetes services. The infrastructure provider can provide a set of additional physical servers to add extra resources to the cluster. However, the total resources are considered to be finite. It is worth mentioning that the same mechanisms for resource profiling and workload prediction are utilized, in a similar way as described above in the architecture of the AHP4HPA framework, but this time they operate for each application individually. Therefore, for each application, there exist distinct resource profiles meaning various flavors of deployed CPU and memory resources. To achieve this, we leverage the Kubernetes resource limits to impose that a running application pod has access to a set of resources, i.e., the maximum CPU and memory respectively. As done in the AHP4HPA, to extract the resource profiles offline benchmarking is assumed by undertaking three different settings, i.e., (i) the request rate of an application, (ii) the deployed resources in terms of CPU and memory of pods, and (iii) the number of replicas for each pod. In such a way, by a simple linear regression method, it is possible to identify linear functions to express the maximum request rate per application to the number of replicas of a specific flavor of resources. Again, Prometheus is utilized to monitor the performance metrics of the applications (app metrics) and the incoming workload traffic (traffic metrics). By utilizing the traffic metrics, the same ARIMA-based Workload Predictor component produces an estimation of the incoming requests for the next time slots, for each application.

The scaling decision for each application is dictated by an AHP4HPA component as described previously in the chapter. The key extension to support the multi-application autoscaling within a KEC is the introduction of the *Cluster AHP Autoscaler* (cAHPA), the component that encapsulates the core intelligence of the proposed framework. The cAHPA acts as a re-optimization engine for the scaling decision for each application paving the way for minimizing the power consumption in the KEC. To this end, the most nominated deployments for each application are taken into consideration and a cluster-level decision is dictated in view of finding a balance between the application QoS requirements and the infrastructure provider's cost.

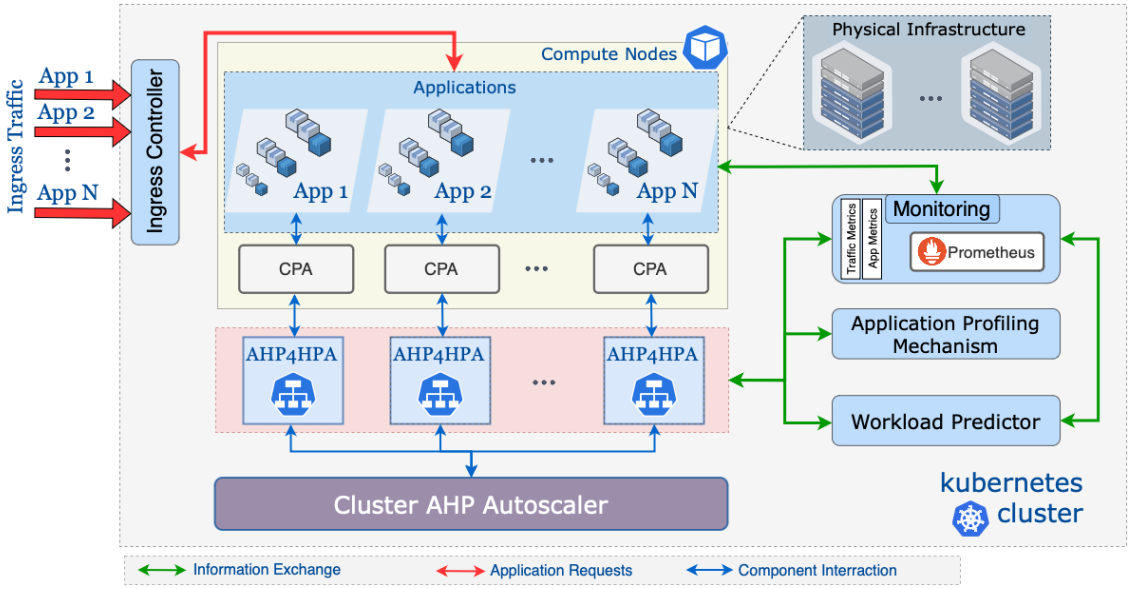


Figure 6.7: Overview architecture of the multi-app framework.

6.6.2 Multi-App Hierarchical Autoscaling

In light of the previously mentioned points, the proposed autoscaling framework for multiple applications within a KEC, is presented. This framework aims to provide the scaling decision for every individual application, within the cluster, in a unified manner; that is, to consolidate the scaling demands of the applications with the optimized resource orchestration of the infrastructure. Specifically, the objectives of cAHPA are efficient resource allocation and power consumption mitigation within the KEC. Starting from the application level, a hierarchical workflow is followed, where cluster candidate deployments are produced based on the deployments of the individual applications. Based on aggregated KPIs, the cAHPA mechanism undertakes the unified scaling decision. Below, the details of the hierarchical framework are presented.

6.6.2.1 Application-level Autoscaling

The AHP4HPA autoscaling mechanism is utilized to provide scaling decisions per application based on the corresponding workload demand. Furthermore, the consolidation of resource demands among several applications has to be enabled. The AHP4HPA mechanism is deployed to undertake the autoscaling process at the application level, as it is able to provide a ranking for alternative scaling decisions of an application. The defined hierarchical structure of Fig. 6.3, containing the respective KPIs of equations (6.1) to (6.10), is taken into account for the AHP algorithm, to provide application-level deployments ranking for scaling. In this framework, based on these KPIs, the output of the AHP4HPA per application is leveraged to produce aggregated candidate deployments for all the applications running in the cluster. In more detail, the AHP4HPA framework

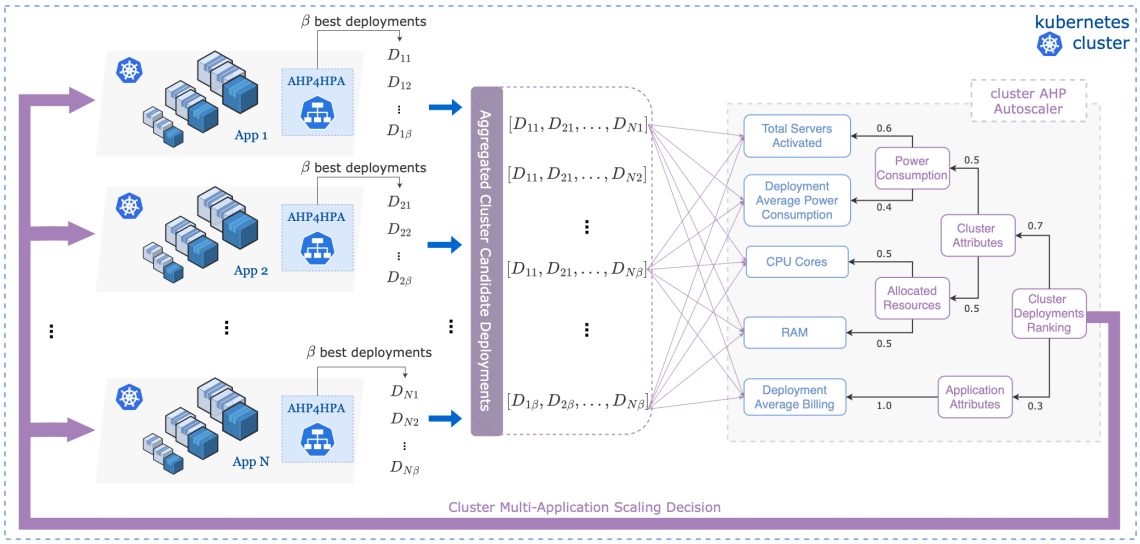


Figure 6.8: Hierarchical Multi-Application Scaling Decision Making.

produces a ranking of the among the candidate deployments $d_{\xi j}$, of an application ξ, using the AHP approach. Taking into account the considered criteria in the AHP4HPA calculations, and the ranking results per application, the adopted policy of the cAHP for providing scaling decisions for multiple applications that are deployed within a KEC is described below.

6.6.2.2 Cluster-level Autoscaling

The cAHPA re-optimizes the allocated resources within the KEC, considering the scaling demands for the respective applications. This process takes place in two phases; the first refers to the definition of the aggregated cluster candidate deployments, and the second regards the multi-application cluster scaling decision.

Cluster Candidate Deployments: The alternative deployments of every application are combined to compose a new set of cluster candidate deployments. Depending on the number of applications, the number of cluster deployments grows significantly, and therefore the complexity of choosing the appropriate candidate also increases. Exploiting the ranking of deployments produced by the AHP4HPA for each application enables the possibility of coping with the increase in complexity. Thus, per application, the β best-ranked, according to the AHP4HPA results, deployments are nominated to compose the cluster candidate solutions. The β parameter is introduced to designate the deployments of each application that are more likely to be part of the overall solution while guaranteeing the performance requirements of the individual applications, which is a hard constraint that the nominated deployments have to meet. Afterward, the nominated deployments of each application are aggregated to produce the cluster candidate deployments. This step of the hierarchical scaling process is illustrated in Fig. 6.8. For N applications, the AHP4HPA

component of each application produces a ranking from which the β best deployments are nominated. For an application ξ the set of nominated deployments is: $\mathcal{D}_\xi = [D_{\xi 1}, D_{\xi 2}, \ldots, D_{\xi\beta}]$, where $D_{\xi i} \in \{d_{\xi j}\}_{j=1}^{\mathcal{M}},\ \forall i \in [1, \ldots, \beta]$. Then, the set of the cluster candidate is the permutations of the sets $\{\mathcal{D}_\xi\}_{\xi=1}^{N}$.

An example of a cluster candidate could be in the form of $[D_{11}, D_{21}, \ldots, D_{N\beta}]$, where the $D_{\xi j}$ is the j^{th} best candidate deployment of the ξ^{th} application. Thus, considering N distinct applications and β best deployments per application, the number of cluster candidates deployments equals β^N.

Cluster AHP-based Scaling Decision: Similarly with the single-application scaling decision-making, the AHP algorithm undertakes the evaluation of the cluster candidate deployments. The cluster's scaling decision concerns the set of deployments that will be nominated for each application. As Fig. 6.8 illustrates, the KPIs are the bottom leaves of the hierarchical tree, while the rest criteria, namely attributes, summarize correlated KPIs. To realize the alignment between the AHP4HPA and the cAHPA mechanism, the selected KPIs reflect similar metrics. Let D^{cl} be a cluster candidate determined as a combination of $D_{\xi j}$ App deployments. The cluster KPIs can be summarized as follows:
(i) the cluster's *Allocated Resources* are determined by the KPIs *CPU Cores* and *RAM*, which are calculated from the aggregated App deployments demands, based on eq. (6.1), (6.2):

$$C^{cl}(D^{cl}) = \sum_{\xi=1}^{\beta} C(D_{\xi j}) \quad (6.14) \quad \text{and} \quad R^{cl}(D^{cl}) = \sum_{\xi=1}^{\beta} R(D_{\xi j}) \quad (6.15)$$

(ii) the *Power Consumption* of the KEC is calculated based on the corresponding KPIs: *Total Active Servers* $S^{cl}(D^{cl})$, and *Deployment Average Power Consumption* $P^{cl}_{avg}(D^{cl})$. These are obtained by substituting the $C(d_{\xi j})$, $R(d_{\xi j})$ for $C^{cl}(D^{cl})$ and $R^{cl}(D^{cl})$ in the equations (6.3) and (6.5).

Based on eq. (6.6), the total power consumption for the cluster candidate is calculated as:

$$P^{cl}_{total}(D^{cl}) = S^{cl}(D^{cl}) * \gamma * P_{MAX} + P^{cl}_{avg}(D^{cl}). \quad (6.16)$$

(iii) the *Average Deployment Billing* for the cluster candidate is calculated based on the billing of one App deployment per application, as computed in eq.(6.9). The average deployment billing of the cluster candidate D^{cl} is calculated as:

$$B^{cl}_{avg}(D^{cl}) = \frac{\sum_{\xi=1}^{\beta} B(D_{\xi j})}{N}. \quad (6.17)$$

Also, in the Fig 6.8, the weight assignment for each decision criterion is noted on the edges of the hierarchical structure, focusing on the trade-off between resource utilization and power consumption of the infrastructure when autoscaling multiple applications. The cluster candidate

Table 6.4: Resource Flavor Setting

	Profiles	Cores	RAM (GB)	Billing ($)	a_i	b_i
App 1	Small	1	2	7	58	-8
	Medium	2	4	13	58.5	3
	Large	4	8	21	78	-12
App 2	Small	0.5	1	3	36	2
	Medium	1	2	5	33	-3
	Large	2	4	13	68	14
App 3	Small	1	2	7	45	-5
	Medium	2	4	13	60	0
	Large	8	16	35	38.87	9
App 4	Small	1	2	7	47	-7
	Medium	2	4	13	71.5	-3
	Large	3	8	17	79.66	11
App 5	Small	1	1	4	30	0
	Medium	1.5	2	7	70	-5
	Large	4	4	17	46.25	15

deployments are matched with values for every KPI. The evaluation of the candidates from the AHP algorithm is performed in a pairwise fashion, as described in 6.4.

6.6.3 A simulation-based evaluation

The evaluation of the proposed framework is achieved via modeling and simulation. Specifically, the proposed framework's efficiency is evaluated and compared against the pure single-application AHP4HPA scaling regarding the allocated CPU Cores and the KEC's power consumption.

6.6.3.1 Experiment Setting

A set of five applications is considered. Concerning Kubernetes deployment parameters, each application is deployed as a set of three discrete resource profiles, namely *small*, *medium*, and *large*, which determine the resource demands. The resources for each profile depend on the characteristics of the application. Table 6.4 includes the resource limits, billing information, and the specifications for the resource profiles per application. The resources per KEC server are 8 CPU cores and 16GB RAM. The maximum number of replicas per resource profile varies from 3 to 5. The maximum power consumption of a KEC server is equal to 2000W, while an idle server consumes 70% of that maximum.

To conduct a fair comparison with the AHP4HPA framework, experimental parameters, which refer to the performance modeling of applications, are determined under the same set of 6.4. Two scaling setups are deployed. In the first, only the AHP4HPA mechanism undertakes the scaling decision per application. On the opposite, the proposed multi-application framework operates to

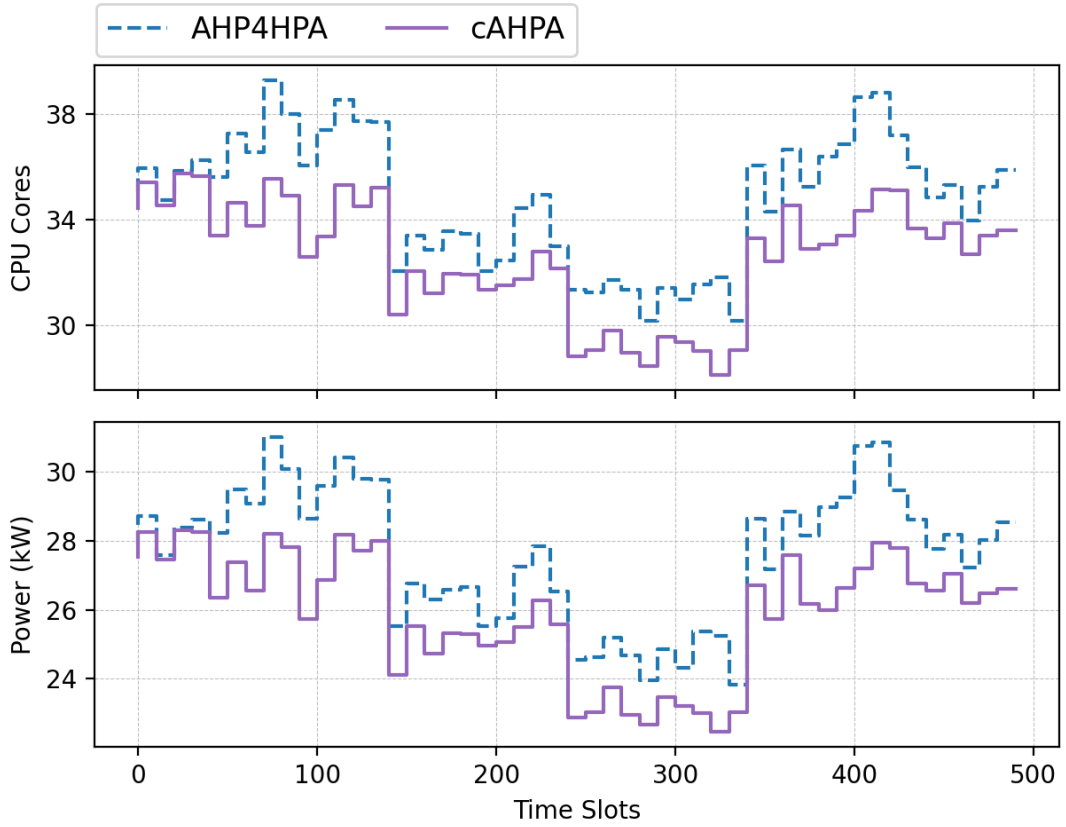


Figure 6.9: CPU Cores and Power Consumption.

produce the scaling decision for every application. The two scaling schemes dictate decisions every 15 seconds for 500 time slots. As mentioned, the parameter β that determines the number of nominated deployments for each application equals 5.

6.6.3.2 Numerical Results

Figure 6.9 illustrates the performance of cAHPA on infrastructure-related metrics. Specifically, the results concern the overall utilization of CPU Cores and the power consumption of the KEC. Regarding the Total allocated CPU cores, the cAHPA achieves an average of 7% reduction compared to the AHP4HPA setting. In periods of high resource demands, the reduction reaches 10% to 12%, which, in absolute numbers, indicates 3.8 less allocated CPU cores. Similar results occurred regarding the KEC's power consumption. In detail, the average mitigation of the consumed power is 6.5%, which is translated to 1670W on average per time slot. During high workload demand periods, the reduction surpasses the 2000W on average, and it is observed in the 28% of the simulation's duration. This result indicates that the proposed hierarchical framework utilizes on average one less server in the whole duration of the experiment. Reducing the utilized CPU

cores and the number of active servers is of particular importance in the context of edge computing with strict infrastructure capacity constraints, and in parallel, leads to a significant mitigation of energy consumption. Finally, it should be stressed that the enhanced performance of the proposed framework against the AHP4HPA mechanism, comes on top of the fact that, as shown in [126], the AHP4HPA mechanism itself already achieves 10% better performance than the standard Kubernetes HPA component in terms of CPU cores and power consumption.

6.7 Chapter Summary

In this part of the thesis, the AHP4HPA framework for autoscaling in Kubernetes Clusters was introduced. Focusing primarily on minimizing power consumption and satisfying several requirements of both infrastructure providers and users, AHP is utilized to enable the ranking of different scaling decisions. Also, a set of KPIs to be included in the hierarchical structure is proposed, as well as the definition of the corresponding weights to adapt the AHP algorithm, in the context of autoscaling with power minimization. To extend the AHP capabilities, the proposed framework includes different resource profiles and a prediction of the incoming workload for each application. Evaluation results indicate a significant reduction in the consumed power from scaling decisions derived from AHP4HPA in comparison with both D-HPA and M-HPA, while reducing the allocated CPU Cores, with a negligible increase in request loss. Furthermore, considering the overall optimization of an infrastructure hosting more than one application, the extension of the framework was considered by designing a hierarchical multi-application scaling decision model, the potential benefits of which were studied using simulations.

Chapter 7

Conclusion and Future Work

This chapter presents the key findings of this thesis in Section 7.1, followed by a summary of the ongoing work and potential avenues for future research 7.2. Additionally, Appendix A provides a list of publications to which the author has contributed throughout the course of this thesis.

7.1 Conclusion

The swift progress of modern IoT applications in the 5G era, accompanied by the incessant surge in resource demand, necessitates the development of intricate approaches to ensure the provision of high-quality services and efficient allocation of resources. This requires close collaboration between infrastructure and service providers to optimize application performance and resource utilization. The challenges associated with optimized resource orchestration in realizing the full potential of the cloud continuum are multifaceted and significant. The dynamic and heterogeneous nature of edge environments, coupled with the diverse requirements of applications, services, and infrastructure providers, necessitates advanced resource allocation and management strategies. Despite the emergence of new architectural concepts, there are numerous challenges that must be addressed to enable seamless management and orchestration of compute and network resources in the Cloud to Edge Continuum.

One of the key challenges is the efficient allocation of resources in edge environments. This involves the selection of appropriate edge clouds based on factors such as proximity, resource availability, and latency requirements. Additionally, the Distributed Virtual Network Embedding problem poses a challenge in mapping VNFs onto the available Edge Cloud infrastructure while considering factors such as network connectivity, resource constraints, and quality of service guarantees.

Another crucial challenge lies in application life-cycle management. This encompasses tasks such as application deployment, scaling, monitoring, and adaptation to dynamic conditions. Effective orchestration mechanisms are needed to automate these processes and optimize resource

allocation based on real-time demands. Additionally, challenges related to service discovery, cross-service communication, delay-aware deployment, reliability, scalability, energy efficiency, and the establishment of network service marketplaces must be addressed to fully harness the potential of the Cloud Continuum.

Addressing these challenges requires ongoing research, collaboration, and standardization efforts. By developing innovative solutions, we can unlock the full potential of the Cloud Continuum, enabling the seamless delivery of advanced applications and services while ensuring optimal performance and resource utilization. Overcoming these challenges will pave the way for a future where the Cloud Continuum operates efficiently, benefiting various industries, businesses, and society as a whole.

Considering the fundamental dimensions in which the aforementioned challenges coexist, it becomes evident that comprehensive and systematic approaches are required to address them effectively. This dissertation discusses specific problems related to the aforementioned challenges. To shed more light on these research topics, Multi-Criteria Decision Analysis techniques are adapted to fit with the problem formulations regarding the edge infrastructure selection for application deployment. Also, the virtual network embedding problem, both within a cloud/edge infrastructure and distributed, among geographically spread data centers is tackled. Several optimization problems were studied, alongside the graph theory, to develop algorithms that are able to provide effective solutions in terms of service provisioning quality, which reflects on network delay minimization. The latter can be accomplished by the proper mapping of VNFs within the servers of an EC for enhanced co-location ratio and elimination in the number of utilized servers and network links, while, in the case of distributed VNE, the round-trip delay minimization, guarantees efficient interactions among VNFs deployed on several layers of the CC, such as the Cloud, the Edge, and the device layer. Another crucial aspect of efficient resource orchestration that this dissertation deals with is the autoscaling of resources, given the dynamic changes in workload demands of emerging 5G, and IoT-related applications. Reactive and proactive scaling methods were leveraged to provide multiple KPIs to be considered, and multi-objective decision-making methods were adopted to develop an autoscaling framework that focuses on resource utilization and energy consumption optimization. It is worth mentioning that the developed mechanisms and algorithms prioritize fast execution to effectively meet the dynamic requirements that arise in the Cloud Continuum ecosystem. Emphasizing the need for timely response and adaptability, the proposed in this thesis approaches are designed to address the dynamic nature of the CC, ensuring that the execution of tasks and services aligns with the evolving conditions and demands of the environment. By prioritizing low computational complexity and responsiveness, the introduced mechanisms and algorithms regarding EC selection and DVNE, aim to deliver efficient and timely outcomes, meeting the real-time needs of the cloud continuum ecosystem.

The aforementioned challenges encompass a range of factors that arise from the demands of modern application development methodologies, encompassing aspects such as dynamic service discovery and the ability to share services. These parameters underscore the need to establish

an architectural ecosystem that can facilitate the dynamic discovery, sharing, and interaction of services [127]. This ecosystem is envisioned as a marketplace, which is a response to the rapid proliferation of services within the Cloud Continuum environment. The proposed methods are subjected to rigorous evaluation through a combination of real-world experimental deployments and comprehensive simulations. This evaluation process aims to assess their reliability and effectiveness, particularly when integrated into industry-adopted architectures and mechanisms for application development and orchestration. The evaluation explores the feasibility of integrating these methods into existing industry frameworks, assessing their compatibility and potential benefits. Based on the comprehensive analysis presented in the preceding chapters for the corresponding problems and challenges, the following research conclusions can be drawn:

- **EC selection is a crucial aspect to enable efficient resource orchestration and application management in the context of Cloud Continuum**. The Cloud Continuum environment comprises a diverse set of computing and network infrastructures, distributed, across different locations. Effective EC selection involves considering factors such as proximity to users, available resources, performance capabilities, and the specific requirements of the application. By selecting the right EC, infrastructure providers, and service tenants can optimize resource utilization, minimize latency, and ensure high-quality service delivery. This dissertation focuses on modeling the EC selection problem and providing a solution that takes into account the heterogeneity of the infrastructures' characteristics and the objectives of the involved stakeholders.

- **Multi-Criteria Decision-Making approaches, are well-suited for addressing the challenges of heterogeneous resource orchestration of ECs.** MCDM methods provide a systematic framework for evaluating and comparing multiple criteria aiming at informed decisions. AHP, in particular, is a widely used MCDM technique that enables assessing the relative importance of various criteria and alternatives in a hierarchical structure. It allows for the quantification of subjective preferences and the calculation of priority weights, facilitating the selection of the most suitable ECs based on multiple objectives and constraints. The use of AHP empowers the involved stakeholders to navigate the complex landscape of EC selection, providing a flexible setting, which enables modifications to support distinct scenarios, by reconstructing its hierarchical structure in terms of the considered criteria, objectives, and weight assignment, to adapt with several, usually, competing objectives of infrastructure providers and applications' users, considering factors such as service performance, cost, proximity, and reliability.

- **Cross-Service Communication establishment could lead to enhanced functionality and improved service delivery.** CSC assumes a pivotal role in enabling the creation of innovative applications that harness the collective capabilities of multiple services. This becomes particularly crucial in emerging domains such as the IoT or Industry 4.0, where timely data exchange is paramount for task execution and workflow optimization. By integrating

CSC functionalities into virtualization technologies and orchestration frameworks, several benefits arise, including enhanced resource orchestration efficiency, streamlined communication between network services, improved resource allocation, and heightened performance optimization. Moreover, the incorporation of CSC promotes the adoption of **Network Service Marketplaces**, facilitating secure and policy-compliant sharing of VNFs among multiple tenants. Integrating CSC into existing research challenges about resource orchestration signifies a vital step towards advancing the state of the art, empowering researchers to address the intricacies of service composition, resource allocation, and performance optimization within the Cloud Continuum environment.

- **Distributed Virtual Network Embedding is a key challenge for efficient resource orchestration in the context of Cloud Continuum.** Recent applications tend to be deployed as interconnected VNFs in a form of a Service Function Chain. Considering the strict capacity constraints of small scaled and geographically spread computing infrastructures, distributing the VNFs across multiple ECs, could lead to optimal resource utilization, and load balancing while improving the overall performance and scalability of the network services. Approaches regarding solving this problem should adapt to meet the requirements of recent IoT applications, for real-time optimization of the VNF embedding, which occurs in the dynamic conditions of the CC ecosystem. The NP-hard nature of the problem further adds to its complexity, necessitating the development of sophisticated algorithms and optimization techniques to find near-optimal or optimal solutions within a reasonable time frame [128]. In this dissertation, the proposed solutions strive to approach this problem by aiming at both providing near-optimal solutions, while eliminating computational complexity of the developed algorithms.

- **Round-trip Delay minimization is of major importance to support emerging application requirements**. The round-trip delay is the dominant performance metric in edge computing systems, especially for QoS guarantee of real-time applications related to IoT [58]. Minimizing the round-trip delay ensures that the applications meet their performance requirements and provide a satisfactory user experience, as the limited computing capabilities of IoT devices require a lot of tasks to be performed fully on edge computing infrastructures or partially via computational offloading. With this capacity, this thesis aims to prioritize round-trip delay minimization, especially in the DVNE problem with hSFC characteristics.

- **Extending existing orchestration frameworks with the above mechanisms benefits both applications and infrastructures**. By incorporating functionalities such as CSC, and low-complexity and efficient algorithms for solving resource selection and VNE problems, emerging 5G applications could gain enhanced quality-of-service, flexibility and interoperability, while infrastructures benefit from improved resource allocation and management, ensuring optimal utilization of available resources and enhancing overall system performance. Thus, in this thesis, proposes a holistic architecture. Edge infrastructure se-

lection, CSC and VNF placement mechanisms are integrated and evaluated to support the deployment of industrial-specific interconnected VNFs as network slices, in a Warehouse Robotics scenario, achieving significant improvements in the expected application QoS and network performance compared to other solutions.

- **Combining reactive and proactive autoscaling KPIs could optimize both application's QoS and infrastructure's operational cost.** Incorporating multiple criteria into the scaling decision process offers significant advantages over approaches reliant solely on threshold values. This comprehensive approach combines proactive elements like workload forecasting with reactive aspects involving various parameter monitoring, all facilitated by multi-criteria decision algorithms, such as AHP. Under this setting, this dissertation, proposes a comprehensive framework that leverages the AHP algorithm's low complexity, coupled with its parameterization flexibility across various levels, to enable the utilization of several conflicting criteria to undertake rapid autoscaling decision-making. The framework, specifically, aiming at the minimization of the resource utilization and the power consumption of the infrastructure perspective, while guaranteeing the applications QoS. Its capability is vital for ensuring timely adaptations of applications' allocated resources aligned with the changing dynamics of the CC environment.

7.2 Future Work

In conclusion, this section highlights potential avenues for future research based on the findings and challenges encountered in this thesis. While the thesis addresses important aspects such as EC selection, CSC, and DVNE, which are critical operations in resource orchestration and application management within the CC context, there are still ample opportunities for further advancements and innovation in this field.

Regarding the problems that this thesis focuses on, some possible extensions worth noting are the following. Initially, from the perspective of Edge Cloud Selection for enabling the Network Service Marketplaces, except the proposed distributed CSC-based service discovery and the MCDM algorithm for the evaluation of the ECs, future research directions in this area include exploring the integration of Blockchain technology for distributed service discovery in multi-domain environments. This can enhance security, transparency, and trust in the service discovery process. Additionally, machine learning methods can be leveraged to develop more accurate and scalable Virtual Edge Cloud profiles, tailored to the specific characteristics and requirements of different marketplaces and driven by fine-grained application profiling. This can lead to improved resource allocation and decision-making in the dynamic and heterogeneous environments of NSMs.

Concerning the DVNE problem, emphasis was placed on the distributed solution of VNE based on the production of k shortest candidate paths, and an initial VNF mapping of hybrid Service Function Chains. One aspect of future work involves further investigation into the effect of the parameter k in the proposed algorithms. By examining different values of k, we can assess its

impact on the efficiency of the DVNE solution. Additionally, as the proposed algorithms exhibit fast execution times, there is potential for online re-optimization of VNE requests. This opens up opportunities for the inclusion of the aspect of dynamic workload management and scaling for the whole hSFC, where control theory and machine learning techniques can be leveraged to optimize resource allocation and predict workload fluctuations. In addition, considering including load-balancing SFC capabilities to generate multi-paths to serve an SFC workload, the proposed solution could be extended to compute solutions for VNE requests that contain split paths [129].

Taking into account the several aspects that compose the proposed autoscaling solution, possible directions for future work could be the enhancement of resource profiles by leveraging Machine Learning (ML) classification algorithms [130]. In that way, a proactive adjustment of the allocated resources will ensure application resilience, and seamless resource orchestration, by avoiding service downtime, and enabling the capability of re-optimization of the resource utilization from the provider's perspective, leading to reduced operation and energy costs. Furthermore, motivated by the obtained results, further investigation regarding the autoscaling solutions that involve multiple and, possible, heterogeneous applications [131] could be examined. The deployment of multiple applications, which belong to different owners, on a common infrastructure makes the autoscaling problem more complex [60]. Thus, the utilization of the feedback provided by application-based mechanisms, like AHP4HPA, to Reinforcement Learning (RL) Agents could enable the global optimization of the whole infrastructure, and applications' performance. Additional benefits could be obtained by the provisioning of an enhanced automated synergistic framework that should place, scale, and migrate containerized applications over distributed infrastructures towards guaranteeing users' requirements and optimizing infrastructure providers' operational costs.

Moreover, in the context of enabling efficient orchestration in the CC to support emerging IoT applications, the exploration of holistic architectural approaches becomes imperative. Additional intriguing avenues for future research encompass the distributed management of end-user mobility, the orchestration of computing resources available at the device layer to optimize resource utilization through the utilization of computational offloading strategies [132], and the integration of trust management mechanisms, such as Service Level Agreement enforcement, to ensure reliable and secure interactions within the Cloud Continuum [40]. Furthermore, the investigation of serverless computing paradigms presents a promising direction for enhancing resource allocation and utilization in a dynamic and scalable manner [40, 133].

Εκτεταμένη Περίληψη στα Ελληνικά

Η ταχεία εξέλιξη της τεχνολογίας ασύρματων δικτύων πέμπτης γενιάς (5G) άνοιξε το δρόμο για μια ποικίλη σειρά καινοτόμων εφαρμογών, η καθεμία από τις οποίες έχει μοναδικές απαιτήσεις, συμπεριλαμβανομένης της χαμηλής καθυστέρησης, της υψηλής αξιοπιστίας και της επεκτασιμότητας, ιδιαίτερα στο πλαίσιο του Διαδικτύου των Αντικειμένων (Internet of Things - IoT). Για την αποτελεσματική υποστήριξη αυτών των εφαρμογών, είναι απαραίτητη η τελεσφόρος διαχείριση και η ενορχήστρωση των πόρων που κατανέμονται σε αυτές σε όλες τις φάσεις του κύκλου ζωής τους. Το πρότυπο διαχείρισης εφαρμογών σε *όλο το φάσμα του υπολογιστικού σύννεφου* (Cloud Continuum - CC), που περιλαμβάνει ένα συνεχές χώρο υπολογιστικών και δικτυακών ετερογενών υποδομών, από κέντρα δεδομένων σύννεφου (Cloud Datacenters), υπολογιστικές υποδομές στα άκρα του δικτύου (Network Edge), έως τερματικές συσκευές, προσφέρει ένα κατανεμημένο υπολογιστικό και δικτυακό περιβάλλον για να ανταποκριθεί στις συγκεκριμένες ανάγκες της ανάπτυξης εφαρμογών IoT. Ωστόσο, *η διαχείριση και ενορχήστρωση (Management and Orchestration - MANO)* των πόρων σε αυτό το πολύπλοκο περιβάλλον αναδύει σημαντικές προκλήσεις. Ο στόχος αυτής της διατριβής είναι να αντιμετωπίσει αυτές τις προκλήσεις προτείνοντας νέων προσεγγίσεων για τη διαχείριση και την ενορχήστρωση πόρων υπολογιστικής και δικτυακής φύσης σε διάφορα επίπεδα του CC. Αυτές οι προσεγγίσεις διαμορφώνονται σύμφωνα με καθιερωμένα πρότυπα όπως το αυτό της Εικονικοποίησης Λειτουργιών Δικτύου (Network Function Virtualization - NFV) του Ευρωπαϊκού Ινστιτούτου Τηλεπικοινωνιακών Προτύπων (European Telecommunications Standards Institute - ETSI) και επικεντρώνονται σε διάφορες πτυχές του κύκλου ζωής των εφαρμογών με διαφορετικά χαρακτηριστικά, όπως η κεντρικοποιημένη, ή κατανεμημένη ανάπτυξη εφαρμογών και η διαχείριση δυναμικής κατανομής πόρων, με στόχο την επίτευξη ισορροπίας μεταξύ των στόχων όλων των εμπλεκομένων οντοτήτων.

Για να επικεντρωθούμε στην επίτευξη της αποτελεσματικής ανάπτυξης των αναδυόμενων υπηρεσιών του 5G, τη βέλτιστη κατανομή πόρων και τη βελτιωμένη συνολική απόδοση και ποιότητα υπηρεσίας, στη διατριβή αυτή αντιμετωπίζονται τα ακόλουθα προβλήματα:

- Επιλογή Υπολογιστικής Υποδομής για ανάπτυξη εφαρμογής: Αυτό περιλαμβάνει την επιλογή της κατάλληλης υποδομής για την ανάπτυξη υπηρεσίας δικτύου στον πλαίσιο του CC. Πρόβλημα στο οποίο απαιτείται να ληφθούν οι στόχοι τόσο των παρόχων της υποδομής και των υπηρεσιών όσο και των χρηστών που αναπτύσσουν τις εφαρμογές τους. Παράλληλα, δίνεται έμφαση στη δυνατότητα αλληλεπίδρασης μεταξύ διαφορετικών τμημάτων από εφαρμογές διαφορετικών χρηστών, προς την εγκαθίδρυση μιας σύγχρονης αγοράς υπηρεσιών δικτύου (Network Service Marketplace - NSM). Το πρότυπο ανάπτυξης εφαρμογών που μελετάται σε αυτό το πρόβλημα, βασιζόμενο στο

πρότυπο NFV, είναι αυτό του δικτυακού τεμαχισμού (Network Slicing). Προτείνεται μια λύση βασισμένη σε αλγόριθμο πολυκριτηριακής βελτιστοποίησης.

- Κατανεμημένη Ενσωμάτωση Εικονικών Δικτύων: Αυτό το πρόβλημα αφορά την ανάπτυξη εφαρμογών IoT, που αποτελούνται από διασυνδεδεμένες υπηρεσίες σε μορφή αλυσίδας. Στο πλαίσιο του NFV μια υπηρεσία αναπτύσσεται ως *εικονική συνάρτηση δικτύου (Virtual Network Function - VNF)*. Αυτές οι αλυσίδες υπηρεσιών μπορούν να αναπτυχθούν σε κατανεμημένες υποδομές στο πλαίσιο του CC, υπό περιορισμούς στη χρήση υπολογιστικών και δικτυακών πόρων. Μελετάται τόσο η αποτελεσματική τοποθέτηση των VNF εντός των υποδομών, όσο και η βέλτιστη κατανεμημένη ενσωμάτωση όλης της αλυσίδας, εστιάζοντας στην ελαχιστοποίηση της μετ' επιστροφής δικτυακής καθυστέρησης. Προτείνονται δύο αλγόριθμοι, που πετυχαίνουν λύσεις κοντά στη βέλτιστη με χαμηλή πολυπλοκότητα.
- Αυτόματη Κλιμάκωση Πόρων για εφαρμογές στα άκρα του δικτύου: Προκειμένου να δοθεί μια ολοκληρωμένη λύση σχετικά με την ενορχήστρωση πόρων στο πλαίσιο του CC, με μια αποτελεσματική διαχείριση του κύκλου ζωής της εφαρμογής, η δυναμική αυτόματη κλιμάκωση των πόρων αναδεικνύεται ως κρίσιμο στοιχείο. Η ενσωμάτωση των δυνατοτήτων αυτόματης κλιμάκωσης σε λύσεις ενορχήστρωσης πόρων επιτρέπει δυναμικές, σε πραγματικό χρόνο προσαρμογές των εκχωρούμενων πόρων που υποστηρίζουν μια εφαρμογή, ανάλογα με τις διακυμάνσεις του φορτίου και τις μεταβαλλόμενες συνθήκες στις υποδομές του CC. Εστιάζοντας, τόσο στους παρόχους υποδομών, όσο και στους χρήστες των εφαρμογών, στη διατριβή προτείνεται ένα πλαίσιο, που βασίζεται σε αλγόριθμο απόφασης πολλών κριτηρίων, αλλά και στην πρόβλεψη φορτίου, για τη διενέργεια αυτόματης κλιμάκωσης πόρων στοχεύοντας στην ελαχιστοποίηση ενεργειακής κατανάλωσης, στη βελτιστοποίηση των υπολογιστικών πόρων, αλλά και στην εγγύηση υψηλής ποιότητας της υπηρεσίας.

Κεφάλαιο 2

Τα θεμελιώδη στοιχεία των κύριων μεθόδων που χρησιμοποιούνται στην παρούσα διατριβή κατατίθενται στο Κεφάλαιο 2. Οι βασικές έννοιες και μαθηματικές μέθοδοι που περιγράφονται έχουν ιδιαίτερη σημασία για την κατανόηση των μοντελοποιήσεων, αλλά και των προτεινόμενων λύσεων σχετικά με τα προβλήματα που καταπιάνεται η διατριβή. Συγκεκριμένα, στο κεφάλαιο, αρχικά, παρουσιάζονται βασικές έννοιες της μεθόδου πολυκριτηριακής λήψης απόφασης Analytic Hierarchy Process (AHP). Η μέθοδος μοντελοποίησης ενός προβλήματος απόφασης βασισμένο σε πολλά κριτήρια, τα οποία ορίζουν μια ιεραρχική δομή και χαρακτηρίζονται από ποικίλα βάρη κατά τη λήψη απόφασης παρουσιάζεται αναλυτικά. Επιπλέον, δίνονται ορισμοί των βασικών μετρικών της μεθόδου, ενώ περιγράφονται οι διαφορετικές προσεγγίσεις υπολογισμού των διανυσμάτων απόφασης για τα κριτήρια κάθε επιπέδου μιας ιεραρχικής δομής, οι οποίες χρησιμοποιούνται για την

προσέγγιση των προβλημάτων επιλογής υπολογιστικής υποδομής στα άκρα του δικτύου (Edge Cloud Selection) για ανάπτυξη εφαρμογών, και ανάληψης αποφάσεων αυτόματης κλιμάκωσης πόρων για εγκιβωτισμένες εφαρμογές (containerized applications), τα οποία συμπεριλαμβάνουν ποικίλα και αντιπαραθετικά κριτήρια ως προς τη βέλτιστη επιλογή. Παράλληλα, παρατίθενται βασικές έννοιες και ορισμοί από τη Θεωρία Γράφων, για τη χρήση βασικών αλγορίθμων κατά την μοντελοποίηση του προβλήματος κατανεμημένης ενσωμάτωσης εικονικών δικτύων (Distributed Virtual Network Embedding - DVNE), αλλά και για την εύρεση λύσεων για τους σχετικούς στόχους βελτιστοποίησης.

Κεφάλαιο 3

Η ανάπτυξη εφαρμογών 5G σε υπολογιστικές υποδομές άκρων σύννεφου (Edge Clouds - EC) γίνεται εφικτή μέσω της εφαρμογής του προτύπου δικτυακού τεμαχισμού (Network Slicing) και δικτυακών υπηρεσιών (ΔΥ), οι οποίες διασφαλίζουν αποτελεσματικά την απομόνωση μεταξύ των διαφόρων εκμισθωτών *δικτυακών τεμαχίων* (ΔΤ). Ωστόσο, αυτή η συνύπαρξη υπηρεσιών εντός ίδιων EC δημιουργεί προοπτικές για αλληλεπιδράσεις μεταξύ αυτών, όπου ένα ΔΤ μπορεί ενδεχομένως να χρησιμοποιήσει ένα ΔΤ από διαφορετικό Μισθωτή. Η εγκαθίδρυση της διατεμαχιακής επικοινωνίας (CSC) όχι μόνο μπορεί να συμβάλλει στη συνολική βελτίωση της απόδοσης των εφαρμογών, περιλαμβάνοντας πτυχές όπως η ποιότητα της υπηρεσίας και η χαμηλή καθυστέρηση, αλλά εισάγει επίσης τη δυνατότητα οικονομικών κερδών μέσω της εκμίσθωσης υπηρεσιών. Επιπλέον, η κοινή χρήση εικονικών μηχανημάτων υπηρεσιών μεταξύ διαφορετικών οντοτήτων που διαχειρίζονται και αναπτύσσουν ΔΤ, γίνεται όλο και πιο σημαντική. Με τα δεδομένα αυτά τίθενται τα θεμέλια για την ανάπτυξη και εγκαθίδρυση ενός περιβάλλοντος αγοράς υπηρεσιών δικτύου (NSM) επόμενης γενιάς, όπου οι χρήστες έχουν τη δυνατότητα να αναπτύσσουν εφαρμογές ως προσαρμοσμένες αλυσίδες υπηρεσιών και καταναλώνοντας ενδεχομένως τμήματα ΔΤ διαφορετικών μισθωτών με ασφαλή και αποτελεσματικό τρόπο. Συνεπώς, ένα NSM θα πρέπει να προσφέρει μια σειρά αυτοματοποιημένων λειτουργιών τόσο στους παρόχους υπηρεσιών όσο και στους καταναλωτές, ενώ θα υποστηρίζει απρόσκοπτα ολόκληρο τον κύκλο ζωής των ΔΥ, που θα περιλαμβάνει πτυχές όπως η καταχώριση υπηρεσίας, η ανακάλυψη, η παρουσίαση και ο τερματισμός, διασφαλίζοντας παράλληλα τη δημιουργία ασφαλών και αυτοματοποιημένων μηχανισμών CSC. Η συγκεκριμένη διατριβή εστιάζει στα προβλήματα κατανεμημένης αναζήτησης υπηρεσιών CSC, και επιλογής υπολογιστικής υποδομής στα άκρα του δικτύου για ανάπτυξη εφαρμογών με τη μορφή δικτυακού τεμαχίου (Network Slice).

Σε μία γεωγραφική περιοχή υπάρχουν πολλές υποδομές, οι οποίες παρέχονται για την ανάπτυξη υπηρεσιών και φιλοξενούν και συγκεκριμένες υπηρεσίες, οι οποίες είναι διαθέσιμες για διατεμαχιακή επικοινωνία. Ένας Καταναλωτής επιλέγει κάποια από τις διαθέσιμες υποδομές για να αναπτύξει το δικό του δικτυακό τεμάχιο. Ωστόσο, στην περίπτωση που επιθυμεί διατεμαχιακή επικοινωνία CSC με μία ή περισσότερες υπηρεσίες που είναι

ήδη ανεπτυγμένες, οι επιλογές του για ανάπτυξη του τεμαχίου του, περιορίζονται στις υποδομές οι οποίες φιλοξενούν τις συγκεκριμένες εφαρμογές. Επομένως, κρίνεται απαραίτητη η υλοποίηση ενός μηχανισμού αναζήτησης υπηρεσιών διαθέσιμων για CSC σε υποδομές EC. Υπό αυτό το πρίσμα, στο Κεφάλαιο 3 της διατριβής παρουσιάζεται μια λύση κατανεμημένης αναζήτησης, η οποία εισάγει τη χρήση προσωρινής μνήμης Cache για κάθε EC, η οποία εμπεριέχει πληροφορίες σχετικά με τις πιο πρόσφατες διαθέσιμες υπηρεσίες CSC για τους γειτονικούς και μόνο κόμβους. Η προώθηση του αιτήματος γίνεται από κάθε EC βάσει της πληροφορίας στην εκάστοτε μνήμη Cache. Ο μηχανισμός εντοπίζει τα υποψήφια EC για την ανάπτυξη τεμαχίου Καταναλωτή EC. Ο Κόμβος Αναζήτησης (Search Node) ορίζεται ως το EC στο οποίο υποβάλλεται το αρχικό αίτημα για την ανάπτυξη τεμαχίων. Τα επιμέρους στοιχεία που ενσωματώνονται στο Μηχανισμό Κατανεμημένης Αναζήτησης (ΜΚΑ), είναι το Μητρώο Υπηρεσιών (Service Registry) και η προσωρινή μνήμη υπηρεσιών (Cache) και περιέχουν όλες τις απαραίτητες πληροφορίες των υπηρεσιών, οι οποίες είναι διαθέσιμες για CSC. Τα στοιχεία αυτά ενσωματώνονται σε κάθε ένα από τα EC. Όπως φαίνεται στον Πίνακα 3.1, για κάθε κεντρική υπηρεσία, η προσωρινή μνήμη υπηρεσίας έχει σταθερό μέγεθος και πολιτική ενημέρωσης βάσει FIFO. Η μνήμη cache κάθε EC ανανεώνεται σε κάθε αλληλεπίδραση με τα γειτονικά EC κατά την αναζήτηση υπηρεσιών CSC, μετά από ένα αίτημα από ένα Μισθωτή Τεμαχίου. Οι εγγραφές στη μνήμη Cache χρησιμοποιούνται από τον αλγόριθμο προώθησης αιτήματος, τον οποίο ενσωματώνει ο ΜΚΑ. Οι εγγραφές στη μνήμη cache πρέπει να ανανεώνονται, ώστε να δίνουν όσο το δυνατόν ακριβέστερη πληροφορία στον ΜΚΑ, μειώνοντας τις περιπτώσεις αστοχίας (cache miss), που μπορεί να προκύψουν από εγγραφές υπηρεσιών που είναι πλέον μη έγκυρες (out-of-date). Ο αλγόριθμος (algorithm) 1, περιγράφει την πολιτική προώθησης του μηνύματος βασισμένη στην μνήμη cache, ενώ παράλληλα για τον καθορισμό του εύρους αναζήτησης γίνεται ορισμός και της παραμέτρου Time-To-Live (TTL), η οποία καθορίζει το "βάθος" αναζήτησης στο γράφο που αναπαριστά το δίκτυο των ECs.

Η πολιτική προώθησης του αιτήματος που εφαρμόζει ο ΜΚΑ είναι η εξής: (i) Εάν μία ή περισσότερες υπηρεσίες διαθέσιμες για CSC (από τις αιτούμενες) υπάρχουν στην μνήμη cache του τρέχοντος EC (Κόμβος Αναζήτησης) στην εγγραφή ενός ή περισσότερων γειτονικών EC, τότε το μήνυμα-αίτημα προωθείται στα συγκεκριμένα γειτονικά EC. (ii) Σε διαφορετική περίπτωση ο Κόμβος Αναζήτησης προωθεί το μήνυμα σε όλους τους γείτονές του (Flooding). (iii) Σε κάθε προώθηση του μηνύματος-αιτήματος το TTL μειώνεται κατά ένα. (iv) Η διαδικασία αναζήτησης επαναλαμβάνεται, με τους ίδιους κανόνες προώθησης, σε κάθε EC, ώσπου η τιμή του TTL να είναι μηδενική.

Μετά το πέρας της αναζήτησης, είναι πιθανό να προκύψουν περισσότερα από ένα υποψήφια EC για την ανάπτυξη του τεμαχίου με δυνατότητα CSC, ενσωματώνουμε και ένα Μηχανισμό Αξιολόγησης Υποδομής (ΜΑΥ), που βασίζεται σε μία πολυκριτηριακή μέθοδο λήψης αποφάσεων λαμβάνοντας υπόψη τις απαιτήσεις του εκάστοτε Μισθωτή Τεμαχίων, καθώς και βασικούς δείκτες απόδοσης (ΒΔΑ) της κάθε υποδομής, αξιολογεί τις λύσεις

και προκύπτει αυτή που ανταποκρίνεται καλύτερα σε κάθε αίτηση για ανάπτυξη νέου τεμαχίου. Η μέθοδος που χρησιμοποιείται είναι η AHP, στην οποία τα κριτήρια απόφασης δομούνται ιεραρχικά και αποτελούνται από τους ΒΔΑ στα φύλλα του ιεραρχικού δέντρου, και στα χαρακτηριστικά απόφασης, τα οποία συνοψίζουν ΒΔΑ κοινού τύπου ή αναφοράς. Η αξιολόγηση των υποψήφιων EC γίνεται βάσει δεδομένων ΒΔΑ, σχετικών με υπηρεσίες υπολογιστικού νέφους και φαίνονται στον Πίνακα 3.2, ενώ για το συγκεκριμένο πρόβλημα η ιεραρχική δομή που χρησιμοποιείται φαίνεται στο Σχήμα 3.2. Τα κριτήρια της δομής αφορούν τόσο τα αυστηρά κριτήρια (Hard Requirements) για τη διατεμαχιακή επικοινωνία, δηλαδή τη Γεωγραφική Ζώνη του EC (Availability Zone) και ποιες από τις ζητούμενες υπηρεσίες για διατεμαχιακή επικοινωνία φιλοξενεί (CSC-enabled Service), όσο και τους ΒΔΑ που αφορούν μη-αυστηρά CSC κριτήρια (Soft Constraints). Επιπλέον, ο μισθωτής, που αιτείται την ανάπτυξη νέου τεμαχίου, έχει την ευχέρεια να θέσει τιμές βάρους σε ορισμένα κριτήρια της ιεραρχικής δομής. Αυτά αφορούν χαρακτηριστικά υψηλότερου επιπέδου και όχι ΒΔΑ, όπου συνήθως απαιτείται εξειδικευμένη γνώση των μετρικών ώστε να τοποθετηθούν σωστά οι τιμές βάρους. Έτσι, λοιπόν, ένας μισθωτής θέτει τιμές βάρους για να εκφράσει την προτίμησή του σχετικά με τα κριτήρια του τρίτου επιπέδου. Επίσης, η AHP επεκτείνεται εισάγοντας την έννοια του εικονικού υποψήφιου EC (virtual EC, VEC), ως σημείο αναφοράς, το οποίο προσδιορίζει την απόσταση μεταξύ των απαιτήσεων του μισθωτή που καταθέτει το αίτημα και των προσφερόμενων μεγεθών των υποψήφιων EC για τους αντίστοιχους ΒΔΑ. Συνεπώς, οι κατάλληλες τιμές στους αντίστοιχους ΒΔΑ, πρέπει να καθοριστούν σύμφωνα με τις απαιτήσεις του εκάστοτε μισθωτή δικτυακού τεμαχίου. Επομένως, με βάση τις τιμές βάρους, που καθορίζονται από το μισθωτή, επιλέγεται αυτόματα ένα συγκεκριμένο προφίλ VEC. Χρησιμοποιώντας πολλές διαφορετικές εκχωρήσεις τιμών βάρους για τα κριτήρια “Performance”, “Cost”, “Computing Performance” και τον αλγόριθμο συσταδοποίησης k-Means παράγονται τα διαφορετικά προφίλ VEC. Αξίζει να σημειωθεί ότι τα βάρη των υπολοίπων χαρακτηριστικών του ίδιου επιπέδου δε χρειάζεται να συμπεριλαμβάνονται στην συσταδοποίηση (clustering), καθώς είναι συμπληρωματικά ως προς τη μονάδα με τα αντίστοιχα που χρησιμοποιούνται. Έτσι, προέκυψαν έξι διαφορετικές συστάδες που αντιστοιχούν σε προφίλ VEC (Σχήμα 3.3). Με τον τρόπο αυτό, ο VEC λαμβάνει μέρος στη διαδικασία με προκαθορισμένες τιμές για τους ΒΔΑ, οι οποίες αντιστοιχούν με κατάλληλο τρόπο στο προφίλ, που προκύπτει από τις προτιμήσεις του μισθωτή. Η εκπαίδευση του συνόλου δεδομένων τιμών βάρους έγινε σε περιβάλλον MatLab και προέκυψαν τα κεντρικά σημεία των έξι συστάδων (Πίνακας 3.3). Για τον υπολογισμό της αξιολόγησης και, εν τέλει, την επιλογή EC, η διαδικασία ξεκινά από πάνω προς τα κάτω με τον υπολογισμό των διανυσμάτων αξιολόγησης για τους ΒΔΑ, με βάση τις τιμές κάθε υποψήφιου EC και του VEC, όπως αναφέρονται στις εξισώσεις (3.2) και (3.3). Δεδομένων των διανυσμάτων αξιολόγησης (RRV) για τους ΒΔΑ, τα αντίστοιχα διανύσματα των ανωτέρω επιπέδων υπολογίζονται λαμβάνοντας υπόψιν τα βάρη για τα αντίστοιχα κριτήρια που ομαδοποιούνται από ένα άλλο, με την εξίσωση (3.4), έως ότου προκύψει η

συνολική αξιολόγηση στο υψηλότερο επίπεδο της δομής. Το διάνυσμα αξιολόγησης είναι κανονικοποιημένο στη μονάδα. Τα EC με σκορ μεγαλύτερο του VEC, θεωρούνται κατάλληλα για ανάπτυξη τεμαχίου βάσει των απαιτήσεων του μισθωτή. Αυτό με το μεγαλύτερο σκορ επιλέγεται αυτόματα από το ΜΑΥ.

Εν συνεχεία, παρουσιάζονται αποτελέσματα πειραματικής αξιολόγησης του προτεινόμενου πλαισίου αυτόματης ανάπτυξης δικτυακών τεμαχίων με απαίτηση CSC. Αρχικά, αξιολογείται ο ΜΚΑ, σε σύγκριση με σχετική μέθοδο από τη βιβλιογραφία ως προς τη αποτελεσματικότητά του στην αναζήτηση των υπηρεσιών CSC, αλλά και στο κόστος, το οποίο αποτυπώνεται ως χρήση δικτυακών συνδέσεων μεταξύ EC, για διάφορα μεγέθη μνήμης cache. Τα σχήματα 3.4 και 3.5, δείχνουν τα σχετικά αποτελέσματα, όπου αποτυπώνεται η ικανότητα του ΜΚΑ να εντοπίζει τις υπηρεσίες CSC για διάφορες ρυθμίσεις στο μέγεθος της cache, αλλά και της τιμής TTL, μειώνοντας αισθητά το κόστος δικτυακής επικοινωνίας. Μάλιστα, συμπεριλαμβάνοντας και την ανταλλαγή μηνυμάτων για την ανανέωση της μνήμης cache κάθε EC, ο προτεινόμενος ΜΚΑ παράγει κατά μέσο όρο 38% λιγότερα μηνύματα από το πλαίσιο NECOS.

Παράλληλα, συγκρίνουμε τις λύσεις, που προκύπτουν μέσω χρήσης του Flooding και μέσω ΜΚΑ, σε σχέση με το αν σε κάθε αίτημα εντοπίζεται EC, το οποίο ικανοποιεί τις απαιτήσεις του μισθωτή τεμαχίου. Σημειώνεται ότι για κάθε πείραμα το Flooding βρίσκει τουλάχιστον ένα υποψήφιο EC με σκορ μεγαλύτερο από το VEC. Για τα υποψήφια EC, όπως προέκυψαν από τον ΜΚΑ, ο ΜΑΥ παρέχει τουλάχιστον μία λύση που ικανοποιεί τις απαιτήσεις του μισθωτή στο 99,5% των πειραμάτων. Σε αυτό το αποτέλεσμα αντικατοπτρίζεται η ικανότητα του ΜΚΑ να εντοπίζει τις λύσεις στην ίδια Γεωγραφική ζώνη με τον Κόμβο αναζήτησης και ουσιαστικά, σε αυτές τις περιπτώσεις, πετυχαίνει το ίδιο αποτέλεσμα με την άπληστη προσέγγιση Flooding με σημαντικά χαμηλότερη επιβάρυνση στο κόστος της επικοινωνίας. Τέλος, ένα αναλυτικό παράδειγμα των υπολογισμών του ΜΑΥ παρατίθεται, με βάση τις τιμές του Πίνακα 3.4.

Κεφάλαιο 4

Η Ενσωμάτωση Εικονικού Δικτύου (Virtual Network Embedding - VNE) είναι ένα τυπικό πρόβλημα στο πλαίσιο του υπολογιστικού Νέφους και του υπολογισμού στα άκρα του Δικτύου. Στοχεύει στον προσδιορισμό της τοποθέτησης εικονικών υπηρεσιών τόσο εσωτερικά στην υποδομή EC όσο και κατανεμημένα, μεταξύ διαφορετικών EC, κάτω από διάφορους περιορισμούς στη χρήση υπολογιστικών και δικτυακών πόρων. Όσον αφορά την τεχνολογία NFV, μια εικονική υπηρεσία αναφέρεται ως VNF. Οι σύγχρονες εφαρμογές 5G που σχετίζονται με το πρότυπο IoT, αναπτύσσονται ως διασυνδεδεμένα VNF με συγκεκριμένη ακολουθία εκτέλεσης σε μια μορφή αλυσίδας υπηρεσιών (Service Function Chain - SFC) συνθέτοντας ένα εικονικό δίκτυο (VN) και το αποτέλεσμα της εκτέλεσης πρέπει να επιστραφεί στην τελική συσκευή, η οποία είναι η πηγή της δικτυακής κίνησης. Έτσι, η ελαχιστοποίηση της καθυστέρησης μετ' επιστροφής είναι μείζονος σημασίας για τη διασφάλιση

των απαιτήσεων απόδοσης των εφαρμογών που βασίζονται στο πρότυπο IoT. Επιπλέον, σε ένα τέτοιο δυναμικό περιβάλλον, η εκ νέου βελτιστοποίηση του VNE απαιτείται συχνά για να καλύψει τις απαιτήσεις καθυστέρησης δικτύου λόγω της κινητικότητας της τελικής συσκευής και άλλων περιορισμών ενορχήστρωσης και διαχείρισης των διαθέσιμων πόρων υποδομών EC, οι οποίοι συνήθως είναι περιορισμένοι. Σε αυτό το Κεφάλαιο, προτείνουμε μια προσέγγιση για Κατανεμημένη Ενσωμάτωση Εικονικού Δικτύου, ή DVNE, η οποία εστιάζει στην ελαχιστοποίηση της καθυστέρησης μετ' επιστροφής. Συγκεκριμένα, προτείνεται αλγόριθμος για την αρχική τοποθέτηση (μέρους) ενός VN, εσωτερικά μιας υποδομής EC, στοχεύοντας στην αποτελεσματική διαχείριση των διαθέσιμων πόρων, που ουσιαστικά συνδέει κάθε VNF του VN με ένα σύνολο υποψήφιων EC που μπορούν να φιλοξενηθούν. Εν συνεχεία, με βάση την αρχική αυτή αντιστοίχιση, περιγράφεται ένας αποτελεσματικός αλγόριθμος βασισμένος σε προσεγγίσεις συντομότερων μονοπατιών σε γράφους, ονομαζόμενος *kSP-DVNE* για να παρέχει τη λύση DVNE, σε πολυωνυμικό χρόνο.

Η συγκεκριμένη μορφή αλυσίδας υπηρεσιών που μελετάται στο πρόβλημα DVNE, κατά την οποία η κίνηση επιστρέφει στην πηγή (για παράδειγμα μία συσκευή IoT), ονομάζεται υβριδική αλυσίδα υπηρεσιών (hybrid Service Function Chain - hSFC) [46, 97]. Το πρόβλημα μοντελοποιείται με τη χρήση Θεωρίας Γράφων, όπως αναλύεται στο υποκεφάλαιο 4.3. Το δίκτυο από υποδομές EC, αναπαρίσταται ως γράφος $G(V, E)$, όπου V είναι οι υποδομές v_i EC και E οι δικτυακές συνδέσεις (v_i, v_j) μεταξύ αυτών. Από την άλλη, το αίτημα VNE, ως αλυσίδα υπηρεσιών, αναπαρίσταται από το γράφο $G_s(V_s, E_s)$, όπου V_s είναι τα VNF s_i του αιτήματος, και E_s οι εικονικές συνδέσεις (s_i, s_j) μεταξύ τους, οι οποίες ορίζουν και τη σειρά εκτέλεσης των λειτουργιών του. Τα υπόλοιπα χαρακτηριστικά του προβλήματος ορίζονται ως ετικέτες (labels) στους δύο γράφους, ενώ συνοψίζονται στον πίνακα 4.2. Σημειώνεται επίσης, ότι για ένα αίτημα ανάπτυξης VNE, μπορεί να απαιτηθεί η χρήση ενός VNF που είναι διαθέσιμο για χρήση από πολλά αιτήματα, το οποίο ορίζεται ως διαμοιραζόμενο VNF (shared VNF). Η προσέγγιση για την επίλυση του προβλήματος γίνεται σε δύο στάδια: *Στάδιο 1:* η αρχική αντιστοίχιση των VNF ενός αιτήματος εσωτερικά της κάθε υποδομής EC και *Στάδιο 2:* η εύρεση της κατανεμημένης λύσης για το αντίστοιχο αίτημα, για την ελαχιστοποίηση της μετ' επιστροφής δικτυακής καθυστέρησης. Η λύση του προβλήματος $\mathcal{P}$, αντιστοιχεί κάθε ένα από τα VNF του αιτήματος με ένα διακομιστή EC, ενώ συμπεριλαμβάνει και το EC, στο οποίο συνδέεται η τελική συσκευή, που είναι η πηγή της δικτυακής κίνησης για το αντίστοιχο αίτημα.

Εν συνεχεία, το πρόβλημα διατυπώνεται μαθηματικά (παρ. 4.4) με βασικό στόχο την ελαχιστοποίηση της μετ' επιστροφής δικτυακής καθυστέρησης, όπως διατυπώνεται στην εξίσωση $(4.4a)$. Ο περιορισμός $(4.4b)$ υποδεικνύει ότι για κάθε λύση $\mathcal{P}$, κάθε υπηρεσία s_i θα πρέπει να έχει μια μοναδική υποδομή EC v στην οποία φιλοξενείται, για ένα συγκεκριμένο αίτημα VNE. Ο περιορισμός $(4.4c)$ εγγυάται ότι οι ζητούμενοι πόροι από τα VNF δεν υπερβαίνουν τη διαθεσιμότητα πόρων κάθε υποδομής EC, ενώ ο $(4.4d)$ διασφαλίζει την ύπαρξη μιας διαδρομής (δικτυακής σύνδεσης) μεταξύ των EC που συνθέτουν

τη λύση DVNE. Στην ουσία, ο περιορισμός (4.4d) εστιάζει στη συνδεσιμότητα του γραφήματος που αντιστοιχεί στη λύση $\mathcal{P}$. Οι διαδρομές που πρόκειται να χρησιμοποιηθούν στη λύση πρέπει να υπάρχουν, επίσης, στην αναπαράσταση γραφήματος της τοπολογίας του δικτύου G. Στη συνέχεια, η προτεινόμενη λύση περιγράφεται διεξοδικά και αποτελείται από δύο μέρη: *(1)* την αρχική αντιστοίχιση ενός VN εντός μιας υποδομής EC, η οποία παρέχει τη (μερική ή μη) αντιστοίχιση του αιτήματος VNE εντός των EC του που συνθέτουν το Δίκτυο και εγγυάται τον περιορισμό (4.4c) και *(2)* τον κύριο αλγόριθμο βασισμένο σε συντομότερα μονοπάτια, ο οποίος αναλαμβάνει τη σύνθεση μιας λύσης DVNE, με βάση την αρχική αντιστοίχιση VN, με ελαχιστοποίηση της μετ' επιστροφής καθυστέρησης, στοχεύοντας παράλληλα στην αντιμετώπιση της εκθετικής πολυπλοκότητας του προβλήματος, επιτυγχάνοντας γρήγορη απόδοση ως προς το χρόνο εκτέλεσης.

Για την προσέγγιση της αρχικής αντιστοίχισης των VNF ενός αιτήματος VNE, αναπτύχθηκε ευριστικός αλγόριθμος, ο οποίος βασίζεται στην διαμέριση του αρχικού αιτήματος, σε επιμέρους *διαμερίσεις εικονικού δικτύου (ΔΕΔ)*, σύμφωνα με την θέση που κατέχουν τα διαμοιραζόμενα VNF στο αίτημα. Στόχος του αλγορίθμου είναι να παρέχει μια αντιστοίχιση του αιτήματος, σε κάθε μία από τις υποδομές EC, μειώνοντας την κατανάλωση πόρων σε αυτή, ενώ παράλληλα αυξάνει τον λόγο συντοποθέτησης γειτονικών VNF σε κοινούς εξυπηρετητές εντός κάθε EC (pair co-location ratio). Η αντιστοίχιση αυτή, θα χρησιμοποιηθεί για την κατασκευή της κατανεμημένης λύσης DVNE. Θεωρώντας ένα σύνολο από χαρακτηριστικά που καθορίζουν την κάθε υποδομή εσωτερικά, όπως είναι ο αριθμός των εξυπηρετητών εντός αυτών και η τοπολογία της σύνδεσής τους με τους αντίστοιχους υπολογιστικούς και δικτυακούς πόρους, ο αλγόριθμος 2 *αρχικής αντιστοίχισης VN με βάση τα διαμοιραζόμενα VNF (shared VNF-based Virtual Network Mapping, sV-VNM)* βασίζεται και στο ότι τα διαμοιραζόμενα VNF είναι ήδη διαθέσιμα και ανεπτυγμένα σε συγκεκριμένους εξυπηρετητές εντός μίας υποδομής. Συγκεκριμένα, η παράμετρος που καθορίζει εάν ένα VNF μπορεί να αντιστοιχιστεί σε έναν εξυπηρετητή ξ της υποδομής ορίζεται στην εξίσωση (4.5), ενώ τα κριτήρια σχετικά με υπολογιστικούς και δικτυακούς πόρους, που πρέπει να καλύπτει η αντιστοίχιση, περιγράφονται στις εξισώσεις (4.6), (4.7). Σημειώνεται ότι οι παράμετροι αυτοί ελέγχονται για κάθε διαμέριση VN. Συνολικά, η πολιτική που ακολουθεί ο αλγόριθμος 2 είναι η ακόλουθη:

- Πραγματοποιείται διαμέριση του αιτήματος VNE με βάση τη θέση των shared VNF.
- Η αντιστοίχιση ξεκινά από τον εξυπηρετητή του shared VNF και προς τα πίσω, για τα υπόλοιπα VNF της ΔΕΔ.
- Εάν μια ΔΕΔ δεν μπορεί αν αντιστοιχιστεί πλήρως λόγω μη διαθεσιμότητας πόρων, διαμερίζεται εκ νέου όπως περιγράφεται στις γραμμές (32-45)
- Κατά την τοποθέτηση προηγούνται οι εξυπηρετητές των VNF που αντιστοιχήθηκαν στο αμέσως προηγούμενο βήμα, ώστε να επιτευχθεί μεγιστοποίηση του λόγου συντοποθέτησης.

- Οι υπόλοιποι εξυπηρετητές που θα εξεταστούν είναι ταξινομημένοι με βάση τους διαθέσιμους πόρους τους σε αύξουσα σειρά.

Η αντιστοίχιση μιας ΔΕΔ πραγματοποιείται με τη διαδικασία PARTITIONMAPPING() (γραμμές 13-31, algorithm 2). Η διαδικασία αντιστοίχισης ξεκινά από τον εξυπηρετητή του shared VNF, εάν αυτό φιλοξενείται σε αυτό το EC, και ακολουθώντας τη διαδρομή του ΔΕΔ προς τα πίσω (γραμμές 14-15), τα υπόλοιπα VNF τοποθετούνται σε με πολιτική *Worst-Fit* με βάση τις απαιτήσεις τους σε πόρους (γραμμές 16 - 29). Στη γραμμή 19, οι περιορισμοί αντιστοίχισης ελέγχονται για κάθε ΔΕΔ. Επίσης, το Σχήμα 4.5 περιέχει ένα παράδειγμα εφαρμογής του αλγορίθμου αυτού.

Δεδομένης της αρχικής αντιστοίχισης, κάθε VNF ενός αιτήματος έχει ένα σύνολο EC υποψηφίων για να αναπτυχθεί κατά τον σχηματισμό της κατανεμημένης λύσης DVNE. Ο προτεινόμενος αλγόριθμος που αναλαμβάνει τη διαδικασία αυτή, βασίζεται στην εύρεση k-συντομότερων μονοπατιών στο γράφο που αναπαριστά το σύστημα, δημιουργώντας ένα σύνολο υποψήφιων υπο-γράφων/μονοπατιών που μπορεί να εξετάσει γραμμικά. Η προτεινόμενη μέθοδος αρχικά βασίζεται στο σχηματισμό ενός επαυξημένου γράφου (ΕΓ) $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$. Οι κόμβοι του ΕΓ είναι το σύνολο των κόμβων των γράφων του δικτύου και του αιτήματος VNE. Επιπλέον, οι ακμές αυτού συμπεριλαμβάνουν τις ακμές του γράφου του δικτύου, και επιπρόσθετα, τοποθετείται μία ακμή μηδενικού βάρους μεταξύ ενός κόμβου EC και ενός κόμβου VNF, εάν το EC είναι υποψήφιο για την ανάπτυξη του VNF, όπως προέκυψε από τη διαδικασία αρχικής αντιστοίχισης. Αφού κάποια (ή όλα) τα VNF ενός αιτήματος δύναται να τοποθετηθούν στην ίδια υποδομή EC, ισχύει η παραδοχή ότι η καλύτερη λύση για την μετ' επιστροφής καθυστέρηση θα δινόταν αν στο συντομότερο μονοπάτι από τον κόμβο εισαγωγής της κίνησης (όπου η τελική συσκευή είναι συνδεδεμένη) προς το πιο μακρινό VNF μπορούσαν να αντιστοιχιστούν το σύνολο των VNF του αιτήματος VNE. Με βάση αυτή την παρατήρηση σχεδιάστηκε αλγόριθμος, ο οποίος χωρίζει το VN σε δύο μονοπάτια: ένα *μονοπάτι προώθησης εικονικού δικτύου* (ΜΠΕΔ) και ένα *μονοπάτι επιστροφής εικονικού δικτύου* (ΜΕΕΔ). Ωστόσο, αφού κάθε VNF μπορεί να έχει τοποθετηθεί οπουδήποτε και ουσιαστικά η λύση είναι ένας κύκλος στο επαυξημένο γράφο, ορίζουμε και τον όρο VNF στόχο, το οποίο μπορεί να είναι οποιοδήποτε από τα VNF του αιτήματος. Άρα τόσα θα είναι και τα ζευγάρια ΜΠΕΔ και ΜΕΕΔ που σχηματίζονται. Για κάθε VNF στόχο υπολογίζονται τα k συντομότερα μονοπάτια στον ΕΓ $\mathcal{G}_a$ από το EC πηγή. Ταξινομώντας το σύνολο των μονοπατιών με βάση το κόστος τους, το οποίο εκφράζει τη δικτυακή καθυστέρηση, ο αλγόριθμος (algorithm) 3, εξετάζει τα υποψήφια μονοπάτια, για το σχηματισμό λύσης, γραμμικά, ως εξής:

- Για κάθε πιθανό VNF στόχο, ορίζει τα ΜΠΕΔ και ΜΕΕΔ.
- Κάθε υποψήφιο μονοπάτι εξετάζεται αν περιέχει τουλάχιστον ένα EC ώστε κάθε VNF να μπορεί να ενσωματωθεί.
- Σχηματίζεται λύση ξεκινώντας από την ενσωμάτωση του ΜΠΕΔ με αντίθετη φορά, προσπελαύνοντας το μονοπάτι ενσωμάτωσης από το VNF στόχο προς το EC πηγή.

- Στη συνέχεια τοποθετεί με ανάλογο τρόπο τα VNF του ΜΕΕΔ.
- Τέλος υπολογίζεται το κόστος της λύσης ως η συνολική δικτυακή καθυστέρηση.

Βασική στόχευση του αλγορίθμου είναι η ταχύτητα εκτέλεσης, για ένα πρόβλημα υψηλής πολυπλοκότητας, μιας και στο πεδίου της ενορχήστρωσης πόρων σε όλο το φάσμα του υπολογιστικού νέφους, η δυναμική και ταχύτατη αλλαγή των συνθηκών όσον αφορά τη χρήση πόρων και των απαιτήσεων των εφαρμογών και των συσκευών είναι δεδομένη. Η πολυπλοκότητα του προτεινόμενου αλγορίθμου αποτυπώνεται στην παράγραφο 4.5.2.3. Η διαδικασία εξετάζει γραμμικά τα $m \times k$ μονοπάτια που παράγονται ως υποψήφια για το σχηματισμό της λύσης. Οπότε η υπολογιστικά ακριβότερη διαδικασία είναι ο υπολογισμός των k συντομότερων μονοπατιών. Έτσι, ο αλγόριθμος επιτυγχάνει πολυωνυμικό κόστος υπολογισμού της λύσης, ανάλογο του πλήθους των ΕC και της τιμής του k, όπως αποτυπώνεται στην εξίσωση (4.11).

Για την αξιολόγηση της προτεινόμενης λύσης, πραγματοποιήθηκαν συγκριτικά πειράματα με άλλες λύσεις της βιβλιογραφίας για τα αντίστοιχα προβλήματα. Σχετικά με τον αλγόριθμο αρχικής αντιστοίχισης sV-VNM για την τοποθέτηση των VNF εντός μιας υποδομής, πραγματοποιήθηκε σύγκριση αποτελεσμάτων με αντίστοιχο αλγόριθμο που χρησιμοποιείται και αυτός ως αρχική λύση στο σχεδιασμό μιας κατανεμημένης λύσης σε σύστημα δύο επιπέδων. Τα συγκριτικά αποτελέσματα που προκύπτουν (Σχήμα 4.9), αναδεικνύουν ότι η μεικτή πολιτική που ακολουθείται από τον sV-VNM αλγόριθμο σχετικά με την διαμέριση του αιτήματος και την επιλογή του επόμενου εξυπηρετητή για τοποθέτηση των VNF, επιτυγχάνει μείωση στη χρήση υπολογιστικών και δικτυακών πόρων, αυξάνοντας την αποτελεσματικότητα χρήσης των εξυπηρετητών (Σχήμα 4.9a, 4.9b), ενώ αυξάνει τον λόγο συντοποθέτησης των γειτονικών VNF τόσο όταν αυτά είναι προς ανάπτυξη, όσο και όταν στο ζευγάρι υπάρχει και ήδη ανεπτυγμένο, διαμοιραζόμενο VNF (Σχήμα 4.9c, 4.9d). Όσον αφορά τον αλγόριθμο, ο οποίος αναλαμβάνει την κατασκευή της κατανεμημένης ενσωμάτωσης, *kSP-DVNE*, πραγματοποιείται συγκριτική αξιολόγηση με λύση που χρησιμοποιείται για αντίστοιχο πρόβλημα και πετυχαίνει το βέλτιστο αποτέλεσμα, με πολύ υψηλό, όμως χρόνο εκτέλεσης, καθώς και με έναν άπληστο αλγόριθμο. Οι παράμετροι των προσομοιώσεων, φαίνονται στους Πίνακες 4.3, 4.4. Στο πρώτο πείραμα, τα VNF του αιτήματος αντιστοιχούνται με τυχαίο τρόπο στα ΕC του δικτύου, για την διεξαγωγή μιας σύγκρισης με τις παραμέτρους να είναι όσο το δυνατόν πιο κοντά σε αυτές της λύσης HSAG με την οποία πραγματοποιείται η σύγκριση. Τα πειραματικά αποτελέσματα αναδεικνύουν την ικανότητα του αλγορίθμου kSP-DVNE να πετυχαίνει το βέλτιστο ή σχεδόν βέλτιστο αποτέλεσμα μειώνοντας δραστικά το χρόνο εκτέλεσης, και την ίδια στιγμή να είναι ξεκάθαρα πιο αποτελεσματικός από μια άπληστη λύση (Σχήμα 4.10). Συγκεκριμένα, ο αλγόριθμος kSP-DVNE πετυχαίνει γραμμική αύξηση του χρόνου εκτέλεσης σε σχέση με το μέγεθος του αιτήματος, σε σύγκριση με την εκθετική άυξηση χρόνου της λύσης HSAG, που αντιστοιχεί σε 90% μείωση, ενώ παράλληλα η μέση μετ' επιστροφής καθυστέρηση που παράγεται είναι μόλις 1.8% μεγαλύτερη από το βέλτιστο. Επίσης, στο 81% των περιπτώσεων πετυχαί-

νει την ελάχιστη δυνατή λύση. Το δεύτερο πείραμα χρησιμοποιεί τον αλγόριθμο αρχικής αντιστοίχισης sV-VNM για τον προσδιορισμό των υποψηφίων EC πριν το σχηματισμό της κατανεμημένης λύσης DVNE, και εκ νέου, συγκρίνονται οι λύσεις kSP-DVNE και HSAG. Ο συνδυασμός των προτεινόμενων λύσεων βελτιώνει ακόμα περισσότερο την απόδοση του αλγορίθμου kSP-DVNE, όπου πετυχαίνει την ελάχιστη δικτυακή καθυστέρηση στο 95% των αιτημάτων, ενώ η μέση απόκλιση από τη βέλτιστη λύση είναι μόλις 0.7%, με δεδομένη τη πολλή σημαντική μείωση στο χρόνο εκτέλεσης (Σχήμα 4.11). Τέλος, στο Σχήμα 4.12 παρουσιάζονται αποτελέσματα προσομοιώσεων, που διενεργήθηκαν με σκοπό τον εντοπισμό σύνδεσης της σύγκλισης του αλγορίθμου σε διάφορες τοπολογίες δικτύου, σε συνάρτηση με την τιμή της παραμέτρου k, που καθορίζει πόσα μονοπάτια για κάθε VNF του αιτήματος θα υπολογιστούν και θα εξεταστούν κατά το σχηματισμό της λύσης DVNE. Στα αποτελέσματα αποτυπώνεται η συσχέτιση της σύγκλισης του αλγορίθμου με την πυκνότητα του γράφου που αναπαριστά το δίκτυο από EC. Επιπλέον, στο σύνολο των διαφορετικών πειραματικών ρυθμίσεων φαίνεται ότι η καταλληλότερη τιμή του k, όπου η σύγκλιση είναι κοντά στη βέλτιστη τιμή είναι η μέση τιμή του βαθμού του γράφου πολλαπλασιασμένη με το πλήθος των κόμβων αυτού. Με αυτή την τιμή, ο προτεινόμενος αλγόριθμος υπολογίζει τη λύση σε πολυωνυμικό χρόνο, το οποίο αποτελεί και το κύριο πλεονέκτημά του. Η ταχύτητα υπολογισμού της λύσης καθιστά τον αλγόριθμο κατάλληλο για κατανεμημένη ενσωμάτωση εικονικών δικτύων, ιδιαίτερα σε εφαρμογές που απαιτούν εκ νέου βελτιστοποίηση της τοποθέτησης των VNF λόγω των δυναμικά μεταβαλλόμενων συνθηκών, που προκύπτουν από κινητικότητα των τερματικών συσκευών, αυξομειώσεις στο πλήθος των αιτημάτων ανά εφαρμογή και της ανάγκης βέλτιστης χρήσης των περιορισμένων πόρων σε μικρές υποδομές στα άκρα του δικτύου.

Κεφάλαιο 5

Στο κεφάλαιο αυτό περιγράφεται η ενσωμάτωση βασικών λειτουργιών ενορχήστρωσης που περιεγράφηκαν στα προηγούμενα κεφάλαια, σε μια αρχιτεκτονική ευθυγραμμισμένη με το καθιερωμένο πρότυπο NFV του ETSI και του Δικτυακού Τεμαχισμού, που επιτρέπει την ανάπτυξη διασυνδεδεμένων VNF ως δικτυακά τεμάχια, ειδικά για την ανάπτυξη εφαρμογών σχετικών με τη Βιομηχανία 4.0, στα άκρα του δικτύου. Συγκεκριμένα, επιδιώκοντας της καθιέρωση της διατεμαχιακής επικοινωνίας, προτείνεται ένας μηχανισμός ανάπτυξης δικτυακών τεμαχίων με δυνατότητα επικοινωνίας μεταξύ VNF τεμαχίων διαφορετικών μισθωτών-εφαρμογών. Εκτός από την παροχή ασφαλούς επικοινωνίας μεταξύ διαφορετικών τεμαχίων, το CSC οδηγεί στην κατανομή μόνο των απαραίτητων υπολογιστικών και δικτυακών πόρων, κάτι που είναι σημαντικό για τη διαχείριση των περιορισμένων πόρων μιας υπολογιστικής βιομηχανικής υποδομής στα άκρα του δικτύου. Η παρούσα προσέγγισή αξιολογείται μέσω μιας εφαρμογής βιομηχανίας 4.0, που αφορά συγκεκριμένα τον τομέα των υπηρεσιών Warehouse Robotics [63], η οποία απαιτεί τον συντονισμό κινητών ρομποτικών πρακτόρων διαφορετικών μισθωτών στο εργοστάσιο. Τα αποτελέσματα

δείχνουν ότι η χρήση του CSC επιτυγχάνει αλληλεπίδραση χαμηλής καθυστέρησης ενώ μειώνει σημαντικά τον χρόνο ολοκλήρωσης μιας αποστολής.

Το Σχήμα 5.1 απεικονίζει την προτεινόμενη επισκόπηση αρχιτεκτονικής, συμπεριλαμβανομένων όλων των απαραίτητων μεμονωμένων στοιχείων κάθε επιπέδου. Εν ολίγοις, το επίπεδο υποδομής άκρων είναι το σημείο όπου οι ρομποτικές εφαρμογές θα αναπτυχθούν ως δικτυακά τεμάχια. Τα βασικά επίπεδα της αρχιτεκτονικής είναι: *(1)* το επίπεδο του Διαχειριστή Εικονικοποιημένης Υποδομής (Virtualizaed Infrastructure Manager - VIM), υπεύθυνο για τη διαχείριση των φυσικών πόρων μιας υποδομής, *(2)* το επίπεδο Διαχείρισης και Ενορχήστρωσης (MANO), το οποίο αλληλεπιδρά με το επίπεδο VIM για τη διαχείριση των εικονικών πόρων της υποδομής NFV (NFV Infrastructure - NFVI) και *(3)* το επίπεδο ενορχήστρωσης διατεμαχιακής επικοινωνίας (CSC Orchestration Layer), το οποίο περιέχει όλες τις απαραίτητες λειτουργικότητες, που επιτρέπουν την αποτελεσματική και ασφαλή αλληλεπίδραση μεταξύ των διαφορετικών δικτυακών τεμαχίων. Αναλυτικότερα, παρατίθενται οι βασικές λειτουργικότητες του επιπέδου ενορχήστρωσης CSC:

- *Αναζήτηση-Μητρώο Υπηρεσιών*: Αυτό το δομικό στοιχείο διατηρεί τη λίστα των υπηρεσιών με δυνατότητα CSC από διαφορετικούς μισθωτές. Όλα τα απαραίτητα δεδομένα που εκτίθενται από κάθε δικτυακό τεμάχιο περιέχονται σε αντίστοιχη εγγραφή στο Μητρώο. Επίσης, πραγματοποιεί την αναζήτηση υπηρεσιών CSC για την εξυπηρέτηση των αντίστοιχων αιτημάτων ανάπτυξης δικτυακού τεμαχίου με αυτή τη δυνατότητα.
- *Επιλογή Υποδομής στα Άκρα του Δικτύου*: Προσδιορίζει την καταλληλότερη υποδομή αιχμής για την υλοποίηση ενός αιτήματος ανάπτυξης τμημάτων από έναν προμηθευτή. Λαμβάνει υπόψη τόσο τις απαιτήσεις του προμηθευτή, συμπεριλαμβανομένης της συντοποθέτησης υπηρεσιών για αιτήματα CSC, όσο και τα δεδομένα της υποδομής σχετικά με τους συγκεκριμένους Βασικούς Δείκτες Απόδοσης, όπως το ελάχιστο εύρος ζώνης, τη διαθεσιμότητα υπηρεσιών και την ελαστικότητα. Αφού υποβληθεί ένα αίτημα CSC, επιλέγονται υποδομές πολλαπλών άκρων ως υποψήφιες για την ανάπτυξη CSC.
- *Τοποθέτηση δικτυακού τεμαχίου εντός της υποδομής*: Η βέλτιστη τοποθέτηση σε μια υποδομή στα άκρα του δικτύου είναι σημαντική για την περαιτέρω μείωση του κόστους επικοινωνίας που υπεισέρχεται από τις πρόσθετες λειτουργίες CSC (π.χ. διαχείριση VNF, τείχος προστασίας και παρακολούθηση). Μετά τη διαδικασία επιλογής υποδομής, η λειτουργία τοποθέτησης δικτυακού τεμαχίου αλληλεπιδρά με το αντίστοιχο VIM. Ανακτά δεδομένα σχετικά με τις υπηρεσίες που πρόκειται να συνδεθούν μέσω του μηχανισμού CSC, σχετικά με τους συγκεκριμένους διακομιστές των υποδομών (φυσικοί εξυπηρετητές) που φιλοξενούν τις υπηρεσίες και στη συνέχεια εκτελεί την τοποθέτηση των VNF ενός τεμαχίου CSC (CSC-slice) το οποίο ενσωματώνει μηχανισμούς ενορχήστρωσης της διατεμαχιακής επικοινωνίας. Ως εκ τούτου με την συντοποθέτηση των VNF που αλληλεπιδρούν μεταξύ των δικτυακών τεμαχίων,

η κίνηση CSC θα διασχίσει την ελάχιστη φυσική διαδρομή μεταξύ αυτών των VNF μέσω του CSC-slice, ελαχιστοποιώντας την πρόσθετη χρονική επιβάρυνση των λειτουργιών CSC και μειώνοντας τη συνολική καθυστέρηση, σε σύγκριση με μια τυχαία τοποθέτηση εντός υποδομής.

- *Κεντρικός Μεσολαβητής*: Τέλος, αυτό το δομικό στοιχείο είναι υπεύθυνο για το χειρισμό των απαραίτητων αλληλεπιδράσεων μεταξύ του επιπέδου ενορχήστρωσης CSC και του επιπέδου MANO, καθώς και για την παροχή μιας διεπαφής για την υποβολή αιτημάτων ανάπτυξης δικτυακού τεμαχίου και CSC. Ο κεντρικός μεσολαβητής είναι επίσης υπεύθυνος για το συντονισμό των αλληλεπιδράσεων μεταξύ των άλλων στοιχείων της πλατφόρμας, ανάλογα με τον τύπο του αιτήματος ανάπτυξης δικτυακού τεμαχίου (π.χ. ανάπτυξη μεμονωμένου τεμαχίου ή αίτημα CSC).

Για να παρουσιάσουμε ένα συγκεκριμένο παράδειγμα χρήσης διατεμαχιακής επικοινωνίας στο πλαίσιο της Βιομηχανίας 4.0, επιλέξαμε μία εφαρμογή σχετική με διαχείριση λειτουργιών αποθήκης με ρομπότ (Warehouse Robotics). Στο Warehouse Robotics, ένα Αυτοματοποιημένο Σύστημα Αποθήκευσης και Ανάκτησης (Automated Storage and Retrieval System - AS/RS) είναι ιδιαίτερα σημαντικό στην επίτευξη της καθιέρωσης των "*έξυπνων εργοστασίων*". Η εφαρμογή αυτή είναι κατάλληλη για την αξιολόγηση της παραπάνω αρχιτεκτονικής, που στοχεύει και στην καθιέρωση της CSC, καθώς απαιτεί την αποτελεσματική συνύπαρξη και συνεργασία ρομποτικών πρακτόρων, που αναλαμβάνουν διαφορετικές αρμοδιότητες εντός ενός τέτοιου πλαισίου.

Αναλυτικότερα, στην παρούσα δουλειά, αναπτύχθηκαν δύο υπηρεσίες ως δικτυακά τεμάχια, που δραστηριοποιούνται σε ένα χώρο έξυπνου εργοστασίου, (1) ένα δικτυακό τεμάχιο για την υπηρεσία ανάκτησης αντικειμένων (Assets Retrieval Slice) και (2) ένα δικτυακό τεμάχιο για την υπηρεσία αποθήκευσης αντικειμένων (Assets Storage Slice). Η υλοποίηση αυτών των υπηρεσιών με τη μορφή δικτυακών τεμαχίων με τα αντίστοιχα VNF για κάθε περίπτωση φαίνονται στο Σχήμα 5.2. Τα ρομπότ συνδεδεμένα με το δικτυακό τεμάχιο της υπηρεσίας ανάκτησης αντικειμένων είναι υπεύθυνα για την εκφόρτωση στοιχείων από τα σημεία αποθήκευσης στο χώρο έξυπνου εργοστασίου. Η όλη λειτουργία υπαγορεύεται από το VNF *Σχεδιαστή Αποστολής (Mission Planner - MP)*, το οποίο αναθέτει αποστολές στα ρομπότ να πάρουν συγκεκριμένες ποσότητες από το ένα απόθεμα. Το VNF *Ανάκτηση Αντικειμένου (Asset Retrieval - AR)* αναλαμβάνει τα εξής καθήκοντα: *i)* να λαμβάνει τη λίστα των απαιτούμενων αντικειμένων για ανάκτηση, από μια εξωτερική υπηρεσία, *ii)* να διατηρεί μια ενημερωμένη προβολή της βάσης δεδομένων αποθέματος και *iii)* να ενημερώσει το δικτυακό τεμάχιο αποθήκευσης αντικειμένων για πιθανές ελλείψεις αποθέματος, επικαλούμενο τον μηχανισμό CSC. Χωρίς απώλεια γενικότητας, υποθέτουμε ότι τα δεδομένα που ανταλλάσσονται μέσω του μηχανισμού CSC κατά τη διάρκεια της επίκλησης μπορεί να κυμαίνονται από απλές κλήσεις Διεπαφής Προγραμματισμού Εφαρμογών (Application Programming Interface) έως αρχεία εικόνας ή βίντεο, ανάλογα με την υλοποίηση των περιγραφόμενων VNF. Το VNF *Υπηρεσίας Εντοπισμού Θέσης (Localization Service - LOC)*

υπολογίζει συνεχώς την ακριβή θέση του ρομπότ και ενημερώνει τα MP και AR VNF. Αντίστοιχα, τα ρομπότ συνδεδεμένα με το δικτυακό τεμάχιο της υπηρεσίας αποθήκευσης αντικειμένων είναι υπεύθυνα για τη φόρτωση των αντικειμένων στα ράφια αποθήκευσης. Σε αυτήν την περίπτωση, το MP VNF είναι υπεύθυνο για την υπαγόρευση των τροχιών που θα πρέπει να ακολουθήσουν τα ρομπότ για να συμπληρώσουν το απόθεμα συγκεκριμένων αντικειμένων. Το MP καλείται από το VNF *Αποθήκευσης Αντικειμένων (Asset Storage - AS)*, το οποίο ενεργοποιείται από το AR VNF του δικτυακού τεμαχίου AR όταν η διαθεσιμότητα ενός αντικειμένου είναι κάτω από ένα προκαθορισμένο όριο, μέσω του μηχανισμού CSC. Όταν ολοκληρωθεί η αποστολή αποθήκευσης, το AS ενημερώνει ασύγχρονα το AR VNF ότι το συγκεκριμένο αντικείμενο είναι διαθέσιμο ξανά. Το LOC VNF εξυπηρετεί τον ίδιο σκοπό όπως και στην παραπάνω περίπτωση. Η ανάγκη για βελτιστοποίηση στην απόδοση των υπηρεσιών στο πλαίσιο ενός έξυπνου εργοστασίου, ειδικότερα μέσω την συνεργασίας ρομποτικών, εν προκειμένω, οντοτήτων, με διαφορετικά καθήκοντα, καθιστά τη βέλτιστη ενορχήστρωση των αντίστοιχων δικτυακών τεμαχίων που υλοποιούν τις επιμέρους λειτουργίες ιδιαίτερης σημασίας. Συνεπώς, το παρόν σενάριο χρήσης κρίνεται κατάλληλο για την αξιολόγηση των μηχανισμών που συζητήθηκαν παραπάνω, μέσω πραγματικής υλοποίησης και ανάπτυξης των αντίστοιχων VNF και τεμαχίων.

Η πειραματική αξιολόγηση έλαβε χώρα στην υποδομή Edge/Cloud του εργαστηρίου *NETMODE*[1], όπου η παραπάνω περιγεγραμμένη αρχιτεκτονική αναπτύχθηκε με τα αντίστοιχα προγράμματα ενορχήστρωσης, καθώς και επεκτάσεις αυτών αναπτύσσοντας τα απαραίτητα APIs. Στόχος είναι η εξαγωγή συμπερασμάτων σχετικά με τα οφέλη του μηχανισμού CSC, της επιλογής υπολογιστικής υποδομής για ανάπτυξη δικτυακών τεμαχίων, καθώς και της τοποθέτησης των VNF εντός μιας υποδομής με κριτήριο την κοινή χρήση VNF μεταξύ διαφορετικών υπηρεσιών. Με βάση ένα σενάριο όπου δύο ρομπότ πρέπει να συνεργαστούν για τις διαδικασίες AS και AR, εξετάζονται τόσο τα οφέλη σε επίπεδο χρόνου εκπλήρωσης της αποστολής, που αντικατοπτρίζει σε ποιότητα των υπηρεσιών αυτών, όσο και η ταχύτητα αποστολής δεδομένων μεταξύ των VNF που αλληλεπιδρούν, λαμβάνοντας υπόψιν διαφορετικές μεθόδους ανάπτυξης των δικτυακών τεμαχίων, αλλά και ενορχήστρωσης αυτών. Το πείραμα που αφορά τη μέτρηση χρόνου ολοκλήρωσης της αποστολής για διαφορετικό αριθμό από σημεία που απαιτείται να περάσουν τα ρομπότ στο χώρο του έξυπνου εργοστασίου για την αποθήκευση και την αποφόρτωση αντικειμένων, λαμβάνονται υπόψη δύο σενάρια: (1) χρήση CSC για την ολοκλήρωση της αποστολής και (2) απομονωμένη λειτουργία των δύο υπηρεσιών . Το Σχήμα 5.3 αποτυπώνει το όφελος της χρήσης διατεμαχιακής επικοινωνίας, αφού ο χρόνος ολοκλήρωσης της αποστολής μειώνεται από 25% έως 30% σε σχέση με την περίπτωση αυτόνομης λειτουργίας για κάθε υπηρεσία. Το δεύτερο πείραμα στοχεύει στην αξιολόγηση των μηχανισμών ενορχήστρωσης μετρώντας την καθυστέρηση μετάδοσης πληροφορίας μεταξύ των εμπλεκόμενων VNF στο παραπάνω σενάριο με χρήση CSC. Η σύγκριση γίνεται για διαφορετικά μεγέθη αρχείων.

[1] https://www.netmode.ntua.gr/

Σαν μέτρο σύγκρισης χρησιμοποιείται η περίπτωση όπου τα δεδομένα μεταφέρονται μεταξύ δύο VNF μέσω απευθείας σύνδεσης (direct link) χωρίς τη χρήση των μηχανισμών CSC -ένα σενάριο, που για λόγους ασφάλειας και απομόνωσης κάθε δικτυακού τεμαχίου είναι μη ρεαλιστικό σε πραγματικές υποδομές- για να αναδειχθούν στο μέγιστο οι δυνατότητες που προσφέρει κάθε μηχανισμός ενορχήστρωσης. Τα αποτελέσματα συμπυκνώνονται στο Σχήμα 5.4. Την ταχύτερη μετάδοση πληροφορίας μεταξύ των δικτυακών τεμαχίων σε σενάριο με χρήση CSC επιτυγχάνει η περίπτωση όπου τα εμπλεκόμενα VNF των δικτυακών τεμαχίων, αλλά και το VNF ενορχήστρωσης της επικοινωνίας CSC αναπτύσσονται σε κοινή υπολογιστική υποδομή EC, ενώ παράλληλα γίνεται και συντοποθέτηση εντός των ίδιων φυσικών μηχανημάτων-διακομιστών (περίπτωση IntraDC Single Edge Server). Τα αποτελέσματα είναι εμφανώς καλύτερα, τόσο στην περίπτωση όπου οι υπηρεσίες τοποθετούνται σε διαφορετικές υποδομές, όσο και όταν είναι στην ίδια υποδομή, αλλά χωρίς βελτιστοποίηση συντοποθέτησης στα διαδοχικά VNF που αλληλεπιδρούν μεταξύ των AS και AR τεμαχίων και το VNF του τεμαχίου CSC.

Κεφάλαιο 6

Η διαχείριση πόρων για εφαρμογές δικτύων 5G σχετικών με το πρότυπο IoT στο Cloud Continuum είναι μια μεγάλη πρόκληση και πολλές πλατφόρμες ενορχήστρωσης, οι οποίες βασίζονται στην τεχνολογία εικονικοποίησης, παρέχουν λειτουργίες για ολόκληρο τον κύκλο ζωής της εφαρμογής. Για παράδειγμα, το *OpenStack* χρησιμοποιείται ευρέως ως VIM για τη διαχείριση εικονικών μηχανών (VM) σε υποδομές cloud [44]. Επιπλέον, το *Kubernetes* χρησιμοποιείται για την ενορχήστρωση και τη διαχείριση εικονικοποιημένων εφαρμογών σε κιβώτια, τα τμήματα των οποίων αναφέρονται ως *container* [45]. Και οι δύο πλατφόρμες προβλέπουν την αυτοματοποίηση της ανάπτυξης, της κλιμάκωσης, της διαχείρισης και της συντήρησης των εφαρμογών. Επικεντρώνοντας το ενδιαφέρον στην *Αυτόματη Κλιμάκωση (Autoscaling)* πόρων, τόσο το Openstack, όσο και το Kubernetes παρέχουν δυνατότητες *οριζόντιας (horizontal)* και *κατακόρηφης (veritcal)* κλιμάκωσης. Και στις δύο πλατφόρμες, η οριζόντια κλιμάκωση χρησιμοποιείται ευρέως. Αντίθετα, η κατακόρυφη κλιμάκωση απαιτεί την επανεκκίνηση του VM (container), επομένως δεν είναι κατάλληλη για διαχείριση πόρων σε πραγματικό χρόνο. Αυτοί οι μηχανισμοί κλιμάκωσης είναι απλοϊκοί και η απόφαση κλιμάκωσης βασίζεται συνήθως σε απλές μετρήσεις παρακολούθησης, όπως η χρήση της CPU και της μνήμης RAM, χωρίς να λαμβάνονται υπόψη άλλες σημαντικές παράμετροι απόδοσης, όπως το δυναμικός φορτίο αιτημάτων σε κάθε container, ή η κατανάλωση ενέργειας. Η ικανοποίηση των αυστηρών απαιτήσεων των αναδυόμενων εφαρμογών 5G όσον αφορά την καθυστέρηση, την απόδοση και την τοποθεσία απαιτεί τη βελτίωση των υφιστάμενων μηχανισμών κλιμάκωσης για την αντιμετώπιση των υποκείμενων προκλήσεων.

Έτσι, λοιπόν, η παρούσα εργασία αποσκοπεί στο να συμπεριλάβει περισσότερα κριτήρια στην απόφαση κλιμάκωσης με επίκεντρο την μείωση στην κατανάλωση ενέργειας και

την αποτελεσματική διαχείριση στους κατανεμημένους πόρους των υποδομών. Προς αυτή την κατεύθυνση σχεδιάστηκε ένα πλαίσιο προσαρμοσμένο για τον ελεγκτή αυτόματης κλιμάκωσης του Kubernetes, που χρησιμοποιεί τον αλγόριθμο λήψης αποφάσεων πολλαπλών κριτηρίων AHP, λαμβάνοντας υπόψιν πολλαπλούς Βασικούς Δείκτες Απόδοσης και παραμέτρους εφαρμογής. Οι κύριες συνεισφορές του κεφαλαίου είναι οι εξής:

- Η απόφαση κλιμάκωσης βασίζεται σε προφίλ πόρων που παρέχουν μια αντιστοίχιση μεταξύ των μετρήσεων ποιότητας της υπηρεσίας και των υπολογιστικών πόρων για ένα σύνολο από όμοια containers, που στο Kubernetes ορίζονται ως *pod* για την ελαχιστοποίηση των παραβιάσεων στα όρια απόδοσης της εφαρμογής.
- Η απόφαση κλιμάκωσης του προτεινόμενου πλαισίου υπολογίζεται από τον αλγόριθμο της μεθόδου AHP, η οποία είναι μια μέθοδος MCDM και λαμβάνει υπόψη διάφορους ΒΔΑ και παραμέτρους, όπως το εισερχόμενο φορτίο εργασίας για μια εφαρμογή, η κατανάλωση ενέργειας, ο αριθμός ενεργών διακομιστών σε μια υποδομή, καθώς και το χρηματικό κόστος για την ανάπτυξη της αντίστοιχης εφαρμογής. Αυτή η προσέγγιση μπορεί να αξιολογήσει πολλές πιθανές λύσεις αυτόματης κλιμάκωσης σε πραγματικό χρόνο και να καταδείξει την αντιστάθμιση μεταξύ της απόδοσης της εφαρμογής και της ελαχιστοποίησης του κόστους, αφού μπορεί να συμπεριλάβει αντιπαραθετικά κριτήρια στην ιεραρχική δομή απόφασης.
- Ακόμη, παρέχεται και η αξιολόγηση του προτεινόμενου πλαισίου, το οποίο αναπτύσσεται ως μια υπηρεσία επέκτασης των δυνατοτήτων του Kubernetes, σε πραγματική, μικρής κλίμακας υποδομή Edge/Cloud, χρησιμοποιώντας μια εφαρμογή αναγνώρισης εικόνων, που είναι υψηλών απαιτήσεων σε υπολογιστικούς πόρους.

Η αρχιτεκτονική του συστήματος αποτυπώνεται στο Σχήμα 6.1. Οι βασικότερες έννοιες που χρησιμοποιούνται και αναφέρονται τόσο στο πλαίσιο Kubernetes όσο και για το σχεδιασμό της προτεινόμενης λύσης αυτόματης κλιμάκωσης είναι οι παρακάτω:

- Το *Pod* είναι η μικρότερη μονάδα ανάπτυξης για εφαρμογές που αναπτύσσονται ως containers στο οικοσύστημα Kubernetes.
- *Η υπηρεσία (service)* είναι ένας όρος που δηλώνει ένα σύνολο από pods, τα οποία έχουν το ίδιο σύνολο λειτουργιών (εφαρμογές)
- Ο όρος *Ανάπτυξη (Deployment)* αναφέρεται στη διαδικασία που αφορά τη δημιουργία, διαχείριση, τροποποίηση και κλιμάκωση των pods μίας εφαρμογής σε μια υποδομή Kubernetes.

Σε αυτήν την εργασία, θεωρούμε ότι κάθε pod τρέχει μόνο ένα container. Έτσι, μια εφαρμογή αποτελείται από ένα σύνολο αντιγράφων από container, τα οποία αντιστοιχούν σε διαφορετικό προφίλ πόρων. Με βάση αυτό δίνεται ο ορισμός της *Ανάπτυξης Εφαρμογής (App Deployment) ως η έννοια που περιλαμβάνει όλες τις αναπτύξεις container όλων των προφίλ πόρων, που εξυπηρετούν τα αιτήματα μιας συγκεκριμένης εφαρμογής*. Το App

Deployment j καθορίζεται, λοιπόν, από τον αριθμό αντιγράφων k_{ij} για κάθε προφίλ πόρων, ως $D_j = \langle k_{ji} \rangle,\ i = 1, \ldots, m$. Με βάση, λοιπόν, το App Deployemnt μιας εφαρμογής, διαμορφώνεται στη συνέχεια, και το σύνολο των ΒΔΑ, ο σχεδιασμός των οποίων είναι ιδιαίτερης σημασίας για την επίτευξη των στόχων του προτεινόμενου πλαισίου. Για την εκμετάλλευση των προφίλ πόρων που δημιουργούνται για κάθε App Deployment, εφαρμόζουμε ένα Αυτοπαλινδρομικό Μοντέλο Κινητού Μέσου Όρου (ARIMA), το οποίο χρησιμοποιείται ευρέως για την πρόβλεψη χρονοσειρών, για να παρέχει μια πρόβλεψη για το ρυθμό αιτημάτων που θα φτάσουν στην εφαρμογή για την επόμενη χρονική περίοδο. Έτσι, μπορεί να οριστεί ΒΔΑ, ο οποίος, βασιζόμενος στην πρόβλεψη, παρέχει και μία μετρική σχετική με την ποιότητα της υπηρεσίας που αναμένεται να έχει ένα συγκεκριμένο App Deployment, όπως προκύπτει και από το προφίλ πόρων μίας συγκεκριμένης εφαρμογής.

Εν συνεχεία, οι κατάλληλοι για το προτεινόμενο σύστημα ΒΔΑ ορίζονται, και μπορούν να κατηγοριοποιηθούν ως εξής: ΒΔΑ (1) *Κατανομής Πόρων* στις εξισώσεις (6.14) για πυρήνες CPU και (6.15) GB μνήμης RAM, (2) *Ενεργειακής Κατανάλωσης* στις εξισώσεις (6.3) για τους ενεργούς διακομιστές και (6.5)-(6.6) για την μέτρηση κατανάλωσης ενέργειας από τη χρήση των App Deployments αλλά και τους διακομιστές που παραμένουν σε αδράνεια, (3) *Διαχειριστικό Κόστος*, που εκφράζει το κόστος των αλλαγών κλιμάκωσης των App Deployment από την τωρινή κατάσταση στην εξίσωση (6.7), αλλά και (4) ΒΔΑ που αφορούν το χρήστη και συγκεκριμένα τη *Χρέωση* για τα αντίστοιχα App Deployment (6.9), και την αναμενόμενη *Ποιότητα της Υπηρεσίας*, που εκφράζεται ως ο βαθμός δυνατότητας εξυπηρέτησης, που είναι ικανό να παρέχει ένα App Deployment για το αντίστοιχο προβλεπόμενο φορτίο, στην εξίσωση (6.10). Ακολούθως, οι ΒΔΑ κατηγοριοποιούνται και οργανώνονται στην ιεραρχική δομή της μεθόδου AHP, όπως φαίνεται στο Σχήμα 6.3. Με βάση τη συγκεκριμένη δομή, ο αλγόριθμος AHP χρησιμοποιείται για να αξιολογήσει τα υποψήφια App Deployment. Για κάθε ένα ΒΔΑ της δομής, ένα υποψήφιο App Deployment παίρνει μία αντίστοιχη τιμή, η οποία υπολογίζεται από τις παραπάνω εξισώσεις, και φυσικά με βάση τα προφίλ πόρων και την πρόβλεψη φορτίου, αλλά και το τρέχον App Deployment. Οι υπολογισμοί του αλγορίθμου AHP στις εξισώσεις (6.11) και (6.12) έχουν σαν αποτέλεσμα ένα διάνυσμα αξιολόγησης, που περιέχει μία τιμή για κάθε μία υποψήφια λύση, όπου το App Deployment με τη μεγαλύτερη τιμή αξιολόγησης στο διάνυσμα του υψηλότερου επιπέδου είναι και αυτό που θα προωθηθεί στον προσαρμοσμένο μηχανισμό κλιμάκωσης του Kubernetes για να αναπτυχθεί στο επόμενο χρονικό διάστημα.

Το προτεινόμενο πλαίσιο αυτόματης κλιμάκωσης αξιολογείται σε μια μικρής κλίμακας υπολογιστική υποδομή Kubernetes. Για μια εφαρμογή αναγνώρισης εικόνας θεωρούμε τρία διαφορετικά προφίλ πόρων, συνδυασμός των οποίων μπορεί να χρησιμοποιηθεί για την ανάπτυξη της εφαρμογής και αναφέρονται, σύμφωνα με το σύνολο των πόρων που χρησιμοποιούν σε ένα αντίγραφό τους, ως *small*, *medium* και *large*. Η αντιστοίχιση προφίλ και πόρων παρουσιάζεται στον Πίνακα 6.2. Επίσης, στο Σχήμα 6.4 απεικονίζεται ο μέγιστος ρυθμός αιτημάτων, που μπορεί να εξυπηρετηθεί ανά δευτερόλεπτο για την επιλεγ-

μένη εφαρμογή σε σχέση με τον αριθμό από αντίγραφα για κάθε προφίλ πόρων και υπολογίστηκε μέσω γραμμικής παλινδρόμησης. Συγκρίνονται τρεις διαφορετικές ρυθμίσεις. Η πρώτη είναι το προτεινόμενο πλαίσιο, που σημειώνεται ως *AHP4HPA*. Οι δύο επόμενες είναι διαφορετικές εκδόσεις του βασικού μηχανισμού αυτόματης οριζόντιας κλιμάκωσης pod (Horizontal Pod Autoscaler - HPA) του Kubernetes. Συγκεκριμένα, αναφέρονται ως *τροποποιημένο HPA, (M-HPA)* και *προεπιλεγμένο HPA (D-HPA)*. Για το M-HPA, τα προφίλ πόρων λαμβάνονται υπόψη στην απόφαση κλιμάκωσης, μαζί με τη δυναμική εξισορρόπηση φορτίου (load balancing). Το HPA προσπαθεί να στοχεύσει τη χρήση CPU στο 70% για τα αναπτυγμένα pods. Από την άλλη πλευρά, για το D-HPA καμία από τις προτεινόμενες τεχνικές δεν χρησιμοποιείται, επομένως επιλέγουμε να στοχεύσουμε τη χρήση της CPU στο 70%, χρησιμοποιώντας μόνο το προφίλ πόρων medium. Για την παραγωγή αμερόληπτων αποτελεσμάτων, και οι τρεις μηχανισμοί κλιμάκωσης λειτουργούν κάθε 15 δευτερόλεπτα, ενώ τους αξιολογούμε σε σχέση με ένα σύνολο δεδομένων (dataset) κίνησης φόρτου εργασίας τριών ωρών από την ανάλυση αιτημάτων μιας τουριστικής εφαρμογής. Επίσης, το παράθυρο σταθεροποίησης κλιμάκωσης είναι απενεργοποιημένο για τις τρεις ρυθμίσεις. Καθώς η αναπτυγμένη εφαρμογή είναι υψηλής υπολογιστικής απαίτησης, στα ακόλουθα αποτελέσματα, εστιάζουμε στη χρήση της CPU. Στο σχήμα 6.5, παρουσιάζεται η κατανάλωση ενέργειας των πόρων που έχουν αναπτυχθεί για τα τρία πειράματα. Υπολογίσαμε την κατανάλωση ενέργειας χρησιμοποιώντας την εξίσωση (6.6). Τα αποτελέσματα για κάθε πείραμα συνοψίζονται στον Πίνακα 6.3. Συγκεκριμένα, η μέση κατανάλωση ενέργειας σε όλο το πείραμα έχει ως αποτέλεσμα 3.96 kWh κατανάλωση για το πλαίσιο AHP4HPA. Ομοίως, στο Σχήμα 6.6, παρουσιάζεται ο συνολικός αριθμός εικονικών πυρήνων CPU (vCPU) για κάθε ρύθμιση. Για το AHP4HPA, οι πυρήνες vCPU που αναπτύσσονται για όλα τα προφίλ πόρων είναι 15.8 κατά μέσο όρο σε όλο το πείραμα. Η προτεινόμενη λύση υπερτερεί τόσο του M-HPA όσο και του D-HPA, έχοντας 9% και 14% λιγότερη μέση κατανάλωση ενέργειας, αντίστοιχα. Το προτεινόμενο πλαίσιο εξυπηρετεί επαρκώς τον εισερχόμενο φόρτο εργασίας έχοντας απώλεια μόνο στο 1.53% των συνολικών αιτημάτων, ενώ το M-HPA έχει 0.37% και το D-HPA έχει 0.25%. Η διαφορά στη συνολική απώλεια αιτημάτων οφείλεται κυρίως στους λανθασμένους υπολογισμούς πρόβλεψης φόρτου εργασίας που παράγονται από την πρόβλεψη του μοντέλου ARIMA. Θα πρέπει, επίσης, να αναφέρουμε ότι η μέση κατανάλωση ενέργειας του M-HPA είναι 4.32 kWh με μέσους χρησιμοποιημένους πυρήνες vCPU 20.5, ενώ του D-HPA είναι 4.57 kWh και 19.4 πυρήνες vCPU.

Τέλος, στο κεφάλαιο αυτό, παρουσιάζεται και μία αρχιτεκτονική, που λαμβάνει αποφάσεις κλιμάκωσης για πολλές εφαρμογές, ταυτόχρονα, οι οποίες είναι ανεπτυγμένες στην ίδια υποδομή. Αυτή η αρχιτεκτονική στοχεύει στο να μπορέσει να πετύχει ακόμα καλύτερα αποτελέσματα στη διαχείριση των πόρων, αλλά και στην ενεργειακή κατανάλωση μια υποδομής, με τη λογική ότι η απόφαση που παίρνουν οι καθιερωμένοι μηχανισμοί, που παρακολουθούν μόνο μία εφαρμογή, δεν λαμβάνουν υπόψη τις απαιτήσεις που έχουν όλες οι ανεπτυγμένες εφαρμογές εντός της ίδιας υποδομής. Συγκεκριμένα, το προτεινό-

μενο πλαίσιο αυτόματης κλιμάκωσης συνολικά για μια υποδομή Kubernetes (Kubernetes Cluster), το οποίο αναφέρεται ως (cluster AHP Autoscaler - cAHPA), ευθυγραμμίζεται με την αρχιτεκτονική του Kubernetes, όπως παρουσιάζεται στο Σχήμα 6.7. Ακολουθεί ιεραρχικό μοντέλο απόφασης. Πρώτα οι μηχανισμοί AHP4HPA, που εξυπηρετούν από μία εφαρμογή αντίστοιχα, αξιολογούν τα App Deployment για κάθε μία από αυτές. Εν συνεχεία, ο cAHPA εξετάζει του πιθανούς συνδυασμούς από App Deployment, που προκρίθηκαν από την αξιολόγηση του AHP4HPA για όλες τις εφαρμογές και, συνολικά, υπολογίζει την καλύτερη λύση. Η λειτουργία του βασίζεται στον ίδιο αλγόριθμο, AHP, με τον τοπικό μηχανισμό AHP4HPA. Το μοντέλο ιεραρχικής απόφασης αυτόματης κλιμάκωσης για όλη την υποδομή φαίνεται στο Σχήμα 6.8. Αρχικά, οι τοπικοί μηχανισμοί AHP4HPA παρέχουν τα καλύτερα App Deployment με βάση την αξιολόγηση που είχαν. Δεν μεταφέρουν όλες τις πιθανές λύσεις, αλλά μόνο αυτές με την υψηλότερη βαθμολογία, που μπορούν, πιθανότερα, να καλύψουν το φορτίο από αιτήματα. Ο γραμμικός συνδυασμός αυτών των App Deployment καθορίζει τις υποψήφιες λύσεις για όλη την υποδομή (μία για κάθε εφαρμογή), που θα εξετάσει ο cAHPA. Η ιεραρχική δομή, η οποία χρησιμοποιείται φαίνεται και αυτή στο Σχήμα 6.8. Οι ΒΔΑ του cAHPA υπολογίζονται συνολικά για την υποδομή με βάση τις εξισώσεις (6.14) έως (6.17), αλλά και τις εξισώσεις του AHP4HPA. Οι αντίστοιχες πράξεις περιγράφονται στην υποενότητα 6.6.2.2. Ακολουθώντας την ίδια λογική με την αξιολόγηση του AHP4HPA, διενεργήθηκαν προσομοιώσεις αξιολόγησης του cAHPA για αυτόματη κλιμάκωση πέντε εφαρμογών, παράλληλα, στην ίδια υποδομή. Τα αποτελέσματα δείχνουν σημαντική βελτίωση στη μέση χρήση πυρήνων vCPU, ειδικά σε χρονικές περιόδου που όλες οι εφαρμογές έχουν υψηλό φορτίο. Συγκεκριμένα η μέση μείωση είναι 7% για το σύνολο των εφαρμογών, ενώ φτάνει έως και 12%, που μεταφράζεται σε 3.8 λιγότερους πυρήνες vCPU, σε σχέση με το όταν οι αποφάσεις κλιμάκωσης παίρνονται αποκλειστικά ανά εφαρμογή. Ανάλογα είναι και τα αποτελέσματα για την ενεργειακή κατανάλωση. Μάλιστα, στο 28% της χρονικής διάρκειας του πειράματος η μείωση της ενεργειακής κατανάλωσης ξεπερνά τα 2000W, που αντιστοιχεί στη μέγιστη ενεργειακή κατανάλωση ενός πλήρως χρησιμοποιούμενου διακομιστή (ενεργός, με 100% χρήση των πυρήνων CPU). Τα αποτελέσματα απεικονίζονται στο Σχήμα 6.9.

Κεφάλαιο 7

Το Κεφάλαιο 7 αναπτύσσεται ως το τελευταίο μέρος αυτής της διατριβής, προσφέροντας μια ολοκληρωμένη σύνθεση των σημαντικότερων αποτελεσμάτων και γνώσεων που εξάγονται από τις ερευνητικές συνεισφορές, όπως αυτές παρουσιάστηκαν παραπάνω. Στο καταληκτικό κεφάλαιο, λοιπόν, τονίζονται τα πιο κομβικά σημεία της έρευνας που πραγματοποιήθηκε σχετικά με μεθόδους βέλτιστης ενορχήστρωσης εφαρμογών δικτύων 5G, σε όλο το φάσμα του υπολογιστικού νέφους. Επιπλέον, τίθενται δρόμοι για πιθανές επεκτάσεις και μελλοντικές εξερευνήσεις, εμπλουτίζοντας τη συνεχιζόμενη συζήτηση σε αυτόν τον τομέα και ενθαρρύνοντας τη συνεχή έρευνα σε σχετικούς τομείς.

Appendix A

Author's Publications

International Peer Reviewed Journals

- **Dimolitsas, I.**, Dechouniotis, D., Papavassiliou, S., Papadimitriou, P. and Theodorou, V., 2021. Edge cloud selection: The essential step for network service marketplaces. IEEE Communications Magazine, 59(10), pp.28-33.

- Laskaratos, D., **Dimolitsas, I.**, Papathanail, G., Xezonaki, M.E., Pentelas, A., Theodorou, V., Dechouniotis, D., Bozios, T., Papadimitriou, P. and Papavassiliou, S., 2022. MESON: A Platform for Optimized Cross-Slice Communication on Edge Computing Infrastructures. IEEE Access, 10, pp.49322-49336.

- **Dimolitsas, I.**, Dechouniotis, D. and Papavassiliou, S., 2023. Time-efficient distributed virtual network embedding for round-trip delay minimization. Journal of Network and Computer Applications, p.103691.

- Filinis N., Tzanettis I., Spatharakis D., Fotopoulou, E., **Dimolitsas, I.**, et al. (2023) "Intent-driven Orchestration of Serverless Applications in the Computing Continuum" Elsevier Future Generation Computer Systems, (Under Review)

- Papadakis-Vlachopapadopoulos, K., **Dimolitsas, I.**, Dechouniotis, D., Tsiropoulou, E.E., Roussaki, I. and Papavassiliou, S., 2021, February. On blockchain-based cross-service communication and resource orchestration on edge clouds. In Informatics (Vol. 8, No. 1, p. 13). MDPI.

- Spatharakis, D., **Dimolitsas, I.**, Dechouniotis, D., Papathanail, G., Fotoglou, I., Papadimitriou, P. and Papavassiliou, S., 2020. A scalable edge computing architecture enabling smart offloading for location based services. Pervasive and Mobile Computing, 67, p.101217.

- Papathanail, G., Pentelas, A., Fotoglou, I., Papadimitriou, P., Katsaros, K.V., Theodorou, V., Soursos, S., Spatharakis, D., **Dimolitsas, I.**, Avgeris, M. and Dechouniotis, D., 2020. Meson: Optimized cross-slice communication for edge computing. IEEE Communications Magazine, 58(10), pp.23-28.

- Papadakis-Vlachopapadopoulos, K., González, R.S., **Dimolitsas, I.**, Dechouniotis, D., Ferrer, A.J. and Papavassiliou, S., 2019. Collaborative SLA and reputation-based trust management in cloud federations. Future Generation Computer Systems, 100, pp.498-512.

- Dechouniotis, D., **Dimolitsas, I.**, Papadakis-Vlachopapadopoulos, K. and Papavassiliou, S., 2018. Fuzzy multi-criteria based trust management in heterogeneous federated future internet testbeds. Future Internet, 10(7), p.58.

International Conferences

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

- **Dimolitsas, I.**, Spatharakis, D., Dechouniotis, D., & Papavassiliou, S. (2022, December). AHP4HPA: An AHP-based Autoscaling Framework for Kubernetes Clusters at the Network Edge. In GLOBECOM 2022-2022 IEEE Global Communications Conference (pp. 2566-2571). IEEE. **(best paper award)**

- **Dimolitsas, I.**, Spatharakis, D., Dechouniotis, D., Zafeiropoulos, A. and Papavassiliou, S., 2023, June. Multi-Application Hierarchical Autoscaling for Kubernetes Edge Clusters. In 2023 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 291-296). IEEE.

- Spatharakis, D., **Dimolitsas, I.**, Genovese, G., Tzanettis, I., Filinis, N., Fotopoulou, E., Vassilakis, C., Zafeiropoulos, A., Iera, A., Molinaro, A. and Papavassiliou, S., 2023, June. A Lightweight Software Stack for IoT Interoperability within the Computing Continuum. In 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT) (pp. 715-722). IEEE.

- Spatharakis, D., **Dimolitsas, I.**, Vlahakis, E., Dechouniotis, D., Athanasopoulos, N., & Papavassiliou, S. (2022, October). Distributed Resource Autoscaling in Kubernetes Edge Clusters. In 2022 18th International Conference on Network and Service Management (CNSM) (pp. 163-169). IEEE.

- **Dimolitsas I.**, Avgeris, M., Spatharakis, D., Dechouniotis D., and Papavassiliou S., 2021, September. Enabling industrial network slicing orchestration: A collaborative edge robotics use case. In 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) (IEEE MeditCom2021). IEEE.

- **Dimolitsas, I.**, Spatharakis, D., Dechouniotis, D., & Papavassiliou, S. (2022, September). A Delay-Aware Approach for Distributed Embedding Towards Cross-Slice Communication. In 2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) (pp. 13-18). IEEE.

- Papathanail, G., **Dimolitsas, I.**, Fotoglou, I., Dechouniotis, D., Papavassiliou, S. and Papadimitriou, P., 2022, June. Towards Secure and Optimized Cross-Slice Communication Establishment. In 2022 IEEE 8th International Conference on Network Softwarization (NetSoft) (pp. 127-132). IEEE.

- Papadakis-Vlachopapadopoulos, K., **Dimolitsas, I.**, Dechouniotis, D., Tsiropoulou, E.E., Roussaki, I. and Papavassiliou, S., 2020, December. Blockchain-based slice orchestration for enabling cross-slice communication at the network edge. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 140-147). IEEE.

- Papathanail, G., Fotoglou, I., Demertzis, C., Pentelas, A., Sgouromitis, K., Papadimitriou, P., Spatharakis, D., **Dimolitsas, I.**, Dechouniotis, D. and Papavassiliou, S., 2020, April. COSMOS: An orchestration framework for smart computation offloading in edge clouds. In NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium (pp. 1-6). IEEE.
- Spatharakis, D., Avgeris, M., Kakkavas G., Papadakis-Vlachopapadopoulos k., **Dimolitsas I.**, Dechouniotis D., Karyotis V., and Papavassiliou S., 2021, July. Industrial robotics experimentation over federated next generation internet testbeds. In 2021 IEEE International Mediterranean Conference on communications and Networking (MeditCom): Demo Sessions (IEEE MeditCom 2021 - Demos). IEEE.
- Bitzi, M., Xezonaki, M.E., Theodorou, V., **Dimolitsas, I.**, Dechouniotis, D., Papavassiliou, S., Papathanail, G., Fotoglou, I., Pentelas, A. and Papadimitriou, P., 2021, September. MESON: Optimized cross-slice communication for pervasive virtual CDN services. In 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) (pp. 1-2). IEEE.

Bibliography

[1] Alessio Botta et al. "Integration of cloud computing and internet of things: a survey." In: *Future generation computer systems* 56 (2016), pp. 684–700.

[2] Dinh C Nguyen et al. "6G Internet of Things: A comprehensive survey." In: *IEEE Internet of Things Journal* 9.1 (2021), pp. 359–383.

[3] Ericsson. *Ericsson mobility report - June 2023*. `https://www.ericsson.com/49dd9d/assets/local/reports-papers/mobility-report/documents/2023/ericsson-mobility-report-june-2023.pdf`. (accessed on 20 June 2023). 2023.

[4] Ejaz Ahmed et al. "Network-centric performance analysis of runtime application migration in mobile cloud computing." In: *Simulation Modelling Practice and Theory* 50 (2015), pp. 42–56.

[5] Pasquale Pace et al. "An edge-based architecture to support efficient applications for healthcare industry 4.0." In: *IEEE Transactions on Industrial Informatics* 15.1 (2018), pp. 481–489.

[6] Carlos Bravo and H Bäckström. "Edge computing and deployment strategies for communication service providers." In: *White Paper, Ericsson* (2020).

[7] Wazir Zada Khan et al. "Edge computing: A survey." In: *Future Generation Computer Systems* 97 (2019), pp. 219–235.

[8] Mohammad Babar et al. "Cloudlet computing: recent advances, taxonomy, and challenges." In: *IEEE access* 9 (2021), pp. 29609–29622.

[9] Wei Bao et al. "Follow me fog: Toward seamless handover timing schemes in a fog computing environment." In: *IEEE Communications Magazine* 55.11 (2017), pp. 72–78.

[10] Yun Chao Hu et al. "Mobile edge computing—A key technology towards 5G." In: *ETSI white paper* 11.11 (2015), pp. 1–16.

[11] Nasir Abbas et al. "Mobile edge computing: A survey." In: *IEEE Internet of Things Journal* 5.1 (2017), pp. 450–465.

[12] Tom H Luan et al. "Fog computing: Focusing on mobile users at the edge." In: *arXiv preprint arXiv:1502.01815* (2015).

[13] Bouziane Brik, Pantelis A Frangoudis, and Adlen Ksentini. "Service-oriented MEC applications placement in a federated edge cloud architecture." In: *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE. 2020, pp. 1–6.

[14] George Papathanail et al. "MESON: Optimized cross-slice communication for edge computing." In: *IEEE Communications Magazine* 58.10 (2020), pp. 23–28. DOI: `10.1109/MCOM.001.2000207`.

[15] Konstantinos Papadakis-Vlachopapadopoulos et al. "On blockchain-based cross-service communication and resource orchestration on edge clouds." In: *Informatics* 8.1 (2021), p. 13. DOI: 10.3390/informatics8010013.

[16] Firdose Saeik et al. "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions." In: *Computer Networks* 195 (2021), p. 108177.

[17] Chuan Sun et al. "Task offloading for end-edge-cloud orchestrated computing in mobile networks." In: *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2020, pp. 1–6.

[18] Lina A Haibeh, Mustapha CE Yagoub, and Abdallah Jarray. "A survey on mobile edge computing infrastructure: Design, resource management, and optimization approaches." In: *IEEE Access* 10 (2022), pp. 27591–27610.

[19] *ETSI GS NFV 006 V2.1.1, Architectural Framework Specification*. https://www.etsi.org/deliver/etsi_gs/nfv/001_099/006/02.01.01_60/gs_nfv006v020101p.pdf. (accessed on 7 December 2022). 2021.

[20] Jose Ordonez-Lucena et al. "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges." In: *IEEE Communications Magazine* 55.5 (2017), pp. 80–87.

[21] Michail-Alexandros Kourtis, Michael J McGrath, et al. "T-NOVA: An open-source MANO stack for NFV infrastructures." In: *IEEE Transactions on Network and Service Management* 14.3 (2017), pp. 586–602.

[22] *OPNFV*. https://www.opnfv.org/. Accessed: 31-03-2022.

[23] Dimitrios Laskaratos et al. "MESON: A Platform for Optimized Cross-Slice Communication on Edge Computing Infrastructures." In: *IEEE Access* 10 (2022), pp. 49322–49336. DOI: 10.1109/ACCESS.2022.3171573.

[24] Ioannis Dimolitsas, Dimitrios Dechouniotis, et al. "A Multi-Criteria Decision Making Method for Network Slice Edge Infrastructure Selection." In: *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2020, pp. 1–7.

[25] Konstantinos Katsaros et al. "Meson: Facilitating cross-slice communications for enhanced service delivery at the edge." In.

[26] Margarita Bitzi et al. "MESON: Optimized cross-slice communication for pervasive virtual CDN services." In: *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE. 2021, pp. 1–2.

[27] Stuart Clayman, Augusto V Neto, et al. "The NECOS Approach to End-to-End Cloud-Network Slicing as a Service." In: *IEEE Communications Magazine* (2021).

[28] Ioannis Dimolitsas et al. "Edge Cloud Selection: The Essential Step for Network Service Marketplaces." In: *IEEE Communications Magazine* 59.10 (2021), pp. 28–33. DOI: 10.1109/MCOM.211.2001056.

[29] Konstantinos Samdanis and Tarik Taleb. "The road beyond 5G: A vision and insight of the key technologies." In: *IEEE Network* 34.2 (2020), pp. 135–141.

[30] Emiliano Casalicchio and Stefano Iannucci. "The state-of-the-art in container technologies: Application, orchestration and security." In: *Concurrency and Computation: Practice and Experience* 32.17 (2020), e5668.

[31] Daniel Rosendo et al. "Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review." In: *Journal of Parallel and Distributed Computing* 166 (2022), pp. 71–94.

[32] Luciano Baresi et al. "A unified model for the mobile-edge-cloud continuum." In: *ACM Transactions on Internet Technology (TOIT)* 19.2 (2019), pp. 1–21.

[33] Josef Spillner, Cristian Mateos, and David A Monge. "Faaster, better, cheaper: The prospect of serverless scientific computing and hpc." In: *High Performance Computing: 4th Latin American Conference, CARLA 2017, Buenos Aires, Argentina, and Colonia del Sacramento, Uruguay, September 20-22, 2017, Revised Selected Papers 4*. Springer. 2018, pp. 154–168.

[34] Sergio Moreschini et al. "Cloud Continuum: the definition." In: *IEEE Access* 10 (2022), pp. 131876–131886.

[35] Bin Gao et al. "Winning at the starting line: Joint network selection and service placement for mobile edge computing." In: *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE. 2019, pp. 1459–1467.

[36] Falak Nawaz et al. "An MCDM method for cloud service selection using a Markov chain and the best-worst method." In: *Knowledge-Based Systems* 159 (2018), pp. 120–131.

[37] Jayadev Gyani, Ahsan Ahmed, and Mohd Anul Haq. "MCDM and various prioritization methods in AHP for CSS: A comprehensive review." In: *IEEE Access* 10 (2022), pp. 33492–33511.

[38] Ioakeim Fotoglou et al. "Towards cross-slice communication for enhanced service delivery at the network edge." In: *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2020, pp. 22–28.

[39] Lucas Bondan, Muriel F Franco, et al. "FENDE: marketplace-based distribution, execution, and life cycle management of VNFs." In: *IEEE Communications Magazine* 57.1 (2019), pp. 13–19.

[40] Luiz Bittencourt et al. "The internet of things, fog and cloud continuum: Integration and challenges." In: *Internet of Things* 3 (2018), pp. 134–155.

[41] Redowan Mahmud, Fernando Luiz Koch, and Rajkumar Buyya. "Cloud-fog interoperability in IoT-enabled healthcare solutions." In: *Proceedings of the 19th international conference on distributed computing and networking*. 2018, pp. 1–10.

[42] Ana Juan Ferrer, Joan Manuel Marquès, and Josep Jorba. "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing." In: *ACM Computing Surveys (CSUR)* 51.6 (2019), pp. 1–36.

[43] Aris Leivadeas et al. "VNF placement optimization at the edge and cloud." In: *Future Internet* 11.3 (2019), p. 69.

[44] *OpenStack*. `https://www.openstack.org/`. (accessed on 7 December 2022). 2022.

[45] *Kubernetes*. `https://kubernetes.io/`. (accessed on 7 December 2022). 2022.

[46] Joel Halpern and Carlos Pignataro. *Service function chaining (SFC) architecture*. Tech. rep. 2015.

[47] Jasenka Dizdarević et al. "A survey of communication protocols for Internet of Things and related challenges of fog and cloud computing integration." In: *ACM Computing Surveys (CSUR)* 51.6 (2019), pp. 1–29. DOI: `10.1145/3292674`.

[48] Djuro Mirkovic, Grenville Armitage, and Philip Branch. "A survey of round trip time prediction systems." In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 1758–1776.

[49] Marios Avgeris et al. "ENERDGE: Distributed energy-aware resource allocation at the edge." In: *Sensors* 22.2 (2022), p. 660.

[50] Chenxi Qiu, Haiying Shen, and Liuhua Chen. "Towards green cloud computing: Demand allocation and pricing policies for cloud service brokerage." In: *IEEE Transactions on Big Data* 5.2 (2018), pp. 238–251.

[51] Gang Sun et al. "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks." In: *Future Generation Computer Systems* 91 (2019), pp. 347–360.

[52] Chinmaya Kumar Dehury and Prasan Kumar Sahoo. "DYVINE: Fitness-based dynamic virtual network embedding in cloud computing." In: *IEEE Journal on Selected Areas in Communications* 37.5 (2019), pp. 1029–1045.

[53] George Papathanail et al. "Towards Secure and Optimized Cross-Slice Communication Establishment." In: *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE. 2022, pp. 127–132.

[54] Edward G Coffman, Michael R Garey, and David S Johnson. "Approximation algorithms for bin-packing—an updated survey." In: *Algorithm design for computer system design* (1984), pp. 49–106.

[55] Angelos Pentelas and Panagiotis Papadimitriou. "Network service embedding for cross-service communication." In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2021, pp. 424–430.

[56] Shuya Zheng et al. "Minimizing the Latency of Embedding Dependence-Aware SFCs into MEC Network via Graph Theory." In: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 1–6. DOI: 10.1109/GLOBECOM46510.2021.9685945.

[57] Ioannis Dimolitsas et al. "A Delay-Aware Approach for Distributed Embedding Towards Cross-Slice Communication." In: *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE. 2022, pp. 13–18.

[58] Olena Skarlat et al. "Resource provisioning for IoT services in the fog." In: *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*. IEEE. 2016, pp. 32–39.

[59] Jin Y Yen. "Finding the k shortest loopless paths in a network." In: *management Science* 17.11 (1971), pp. 712–716. DOI: 10.1287/mnsc.17.11.712.

[60] Tarik Taleb, Ibrahim Afolabi, and Miloud Bagaa. "Orchestrating 5G network slices to support industrial internet and to shape next-generation smart factories." In: *Ieee Network* 33.4 (2019), pp. 146–154. DOI: 10.1109/MNET.2018.1800129.

[61] Bufan Liu et al. "Edge-cloud orchestration driven industrial smart product-service systems solution design based on CPS and IIoT." In: *Advanced Engineering Informatics* 42 (2019), p. 100984.

[62] Ioannis Dimolitsas et al. "Enabling industrial network slicing orchestration: A collaborative edge robotics use case." In: *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE. 2021, pp. 215–220.

[63] Tommaso Pardi et al. "A Soft Robotics Approach to Autonomous Warehouse Picking." In: *Advances on Robotic Item Picking*. Springer, 2020, pp. 23–35.

[64] Waheed Iqbal et al. "Predictive Auto-Scaling of Multi-Tier Applications Using Performance Varying Cloud Resources." In: *IEEE Transactions on Cloud Computing* 10.1 (2022), pp. 595–607. DOI: 10.1109/TCC.2019.2944364.

[65] R. Calheiros et al. "Workload prediction using ARIMA model and its impact on cloud applications' QoS." In: *IEEE Transactions on Cloud Computing* 3.4 (2014), pp. 449–458.

[66] José Figueira, Salvatore Greco, and Matthias Ehrgott. "Multiple criteria decision analysis: state of the art surveys." In: (2005).

[67] Sylvain Kubler et al. "A state-of the-art survey & testbed of fuzzy AHP (FAHP) applications." In: *Expert systems with applications* 65 (2016), pp. 398–422.

[68] Ioannis Patiniotakis et al. "Managing imprecise criteria in cloud service ranking with a fuzzy multi-criteria decision making method." In: *Service-Oriented and Cloud Computing: Second European Conference, ESOCC 2013, Málaga, Spain, September 11-13, 2013. Proceedings 2*. Springer. 2013, pp. 34–48.

[69] Matteo Brunelli. *Introduction to the analytic hierarchy process*. Springer, 2014.

[70] Da-Yong Chang. "Applications of the extent analysis method on fuzzy AHP." In: *European journal of operational research* 95.3 (1996), pp. 649–655.

[71] Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya. "SMICloud: A Framework for Comparing and Ranking Cloud Services." In: *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. 2011, pp. 210–218. DOI: 10.1109/UCC.2011.36.

[72] Alessio Ishizaka and Markus Lusti. "How to derive priorities in AHP: a comparative study." In: *Central European Journal of Operations Research* 14 (2006), pp. 387–400.

[73] Gordon Crawford and Cindy Williams. "A note on the analysis of subjective judgment matrices." In: *Journal of mathematical psychology* 29.4 (1985), pp. 387–405.

[74] Roseanna W Saaty. "The analytic hierarchy process—what it is and how it is used." In: *Mathematical modelling* 9.3-5 (1987), pp. 161–176.

[75] Reinhard Diestel and Reinhard Diestel. "Extremal graph theory." In: *Graph theory* (2017), pp. 173–207.

[76] Douglas Brent West et al. *Introduction to graph theory*. Vol. 2. Prentice hall Upper Saddle River, 2001.

[77] Dimitrios Dechouniotis, Nikolaos Athanasopoulos, et al. "Edge Computing Resource Allocation for Dynamic Networks: The DRUID-NET Vision and Perspective." In: *Sensors* 20.8 (2020), p. 2191.

[78] ETSI GS NFV-SOL 006. *Network Functions Virtualisation (NFV); Protocols and Data Models*. 2019. URL: https://www.etsi.org/deliver/etsi%5C_gs/NFV-SOL/001%5C_099/006/02.06.01%5C_60/gs%5C_NFV-SOL006v020601p.pdf.

[79] Raphael Vicente Rosa and Christian Esteve Rothenberg. "The Pandora of Network Slicing: A Multicriteria Analysis." In: *Trans. on Emerging Telecom. Tech.* 31.1 (2020), pp. 1–17.

[80] Spyridon Vassilaras et al. "The algorithmic aspects of network slicing." In: *IEEE Communications Magazine* 55.8 (2017), pp. 112–119.

[81] ETSI GS NFV 002. *Network Functions Virtualisation (NFV); Architectural Framework*. 2013. URL: https://www.etsi.org/deliver/etsi%5C_gs/NFV/001%5C_099/002/01.01.01%5C%5C%5C_60/gs%5C_NFV002v010101p.pdf.

[82] ETSI. *Open Source MANO*. URL: http://osm.etsi.org.

[83] Riccardo Guerzoni, Ishan Vaishnavi, et al. "Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: an architectural survey." In: *Transactions on Emerging Telecommunications Technologies* 28.4 (2017), pp. 1–19.

[84] Gino Carrozzo, M Shuaib Siddiqui, August Betzler, et al. "AI-driven Zero-touch Operations, Security and Trust in Multi-operator 5G Networks: a Conceptual Architecture." In: *IEEE European Conf. on Net. and Comm. (EuCNC)*, pp. 254–258.

[85] Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya. "A framework for ranking of cloud computing services." In: *Future Generation Computer Systems* 29.4 (2013), pp. 1012–1023.

[86] Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya. "Smicloud: A framework for comparing and ranking cloud services." In: *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. IEEE. 2011, pp. 210–218.

[87] Konstantinos Papadakis-Vlachopapadopoulos et al. "Collaborative SLA and reputation-based trust management in cloud federations." In: *Future Generation Computer Systems* 100 (2019), pp. 498–512.

[88] Dimitrios Dechouniotis et al. "Fuzzy multi-criteria based trust management in heterogeneous federated future internet testbeds." In: *Future Internet* 10.7 (2018), p. 58.

[89] Xianrong Zheng et al. "CLOUDQUAL: a quality model for cloud services." In: *IEEE transactions on industrial informatics* 10.2 (2014), pp. 1527–1536.

[90] IEEE. "IEEE Standard for a Software Quality Metrics Methodology." In: *IEEE Std 1061 TM-1998 (R2009)* (2009).

[91] Xiaoyong Li et al. "Service operator-aware trust scheme for resource matchmaking across multiple clouds." In: *IEEE transactions on parallel and distributed systems* 26.5 (2014), pp. 1419–1429.

[92] Jane Siegel and Jeff Perdue. "Cloud services measures for global use: the service measurement index (SMI)." In: *2012 Annual SRII global conference*. IEEE. 2012, pp. 411–415.

[93] Geoff Coyle. "The analytic hierarchy process (AHP)." In: *Practical strategy: Structured tools and techniques* (2004), pp. 1–11.

[94] Stuart Lloyd. "Least squares quantization in PCM." In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[95] The University of Adelaide. *The Internet Topology Zoo*. `http://www.topology-zoo.org/index.html`. (accessed on 7 December 2022). 2013.

[96] Thomas H Cormen, Charles E Leiserson, et al. *Introduction to Algorithms*. MIT press, 2009.

[97] Danyang Zheng et al. "Toward optimal hybrid service function chain embedding in multi-access edge computing." In: *IEEE Internet of Things Journal* 7.7 (2019), pp. 6035–6045. DOI: `10.1109/JIOT.2019.2957961`.

[98] Paola Cappanera, Federica Paganelli, and Francesca Paradiso. "VNF placement for service chaining in a distributed cloud environment with multiple stakeholders." In: *Computer Communications* 133 (2019), pp. 24–40. DOI: `10.1016/j.comcom.2018.10.008`.

[99] Jianing Pei et al. "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system." In: *IEEE Transactions on Parallel and Distributed Systems* 30.10 (2018), pp. 2179–2192. DOI: `10.1109/TPDS.2018.2880992`.

[100] Balázs Németh et al. "Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure." In: *IEEE Transactions on Mobile Computing* 21.9 (2021), pp. 3150–3162. DOI: `10.1109/TMC.2021.3055426`.

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

[102] Dmitrii Chemodanov et al. "A constrained shortest path scheme for virtual network service management." In: *IEEE Transactions on Network and Service Management* 16.1 (2018), pp. 127–142. DOI: 10.1109/TNSM.2018.2865204.

[103] Rong Chai et al. "Multi-objective optimization-based virtual network embedding algorithm for software-defined networking." In: *IEEE Transactions on Network and Service Management* 17.1 (2019), pp. 532–546. DOI: 10.1109/TNSM.2019.2953297.

[104] L. Bondan et al. "FENDE: Marketplace-Based Distribution, Execution, and Life Cycle Management of VNFs." In: *IEEE Communications Magazine* 57.1 (2019), pp. 13–19. DOI: 10.1109/MCOM.2018.1800507.

[105] Matthias Rost and Stefan Schmid. "On the hardness and inapproximability of virtual network embeddings." In: *IEEE/ACM Transactions on Networking* 28.2 (2020), pp. 791–803.

[106] Antonio Marotta et al. "A fast robust optimization-based heuristic for the deployment of green virtual network functions." In: *Journal of Network and Computer Applications* 95 (2017), pp. 42–53. DOI: 10.1016/j.jnca.2017.07.014.

[107] Robert W Floyd. "Algorithm 97: shortest path." In: *Communications of the ACM* 5.6 (1962), p. 345. DOI: 10.1145/367766.368168.

[108] *NetworkX, Network Analysis in Python*. https://networkx.org. (Last accessed on 15/07/23). 2023.

[109] Dimitrios Spatharakis et al. "Industrial Robotics Experimentation over Federated Next Generation Internet Testbeds." In: *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE. 2021, pp. 1–2.

[110] Gajamohan Mohanarajah et al. "Rapyuta: A cloud robotics platform." In: *IEEE Transactions on Automation Science and Engineering* 12.2 (2014), pp. 481–493.

[111] Alessio Botta, Luigi Gallo, and Giorgio Ventre. "Cloud, Fog, and Dew Robotics: Architectures for next generation applications." In: *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE. 2019, pp. 16–23.

[112] Mohammed Salman Shaik et al. "Enabling Fog-based Industrial Robotics Systems." In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE. 2020, pp. 61–68.

[113] Kiril Antevski et al. "On the integration of NFV and MEC technologies: architecture analysis and benefits for edge robotics." In: *Computer Networks* 175 (2020), p. 107274.

[114] Kiril Antevski et al. "DLT federation for Edge robotics." In: *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE. 2020, pp. 71–76.

[115] Marios Avgeris et al. "Single vision-based self-localization for autonomous robotic agents." In: *2019 7th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE. 2019, pp. 123–129.

[116] Dimitrios Spatharakis et al. "A Switching Offloading Mechanism for Path Planning and Localization in Robotic Applications." In: *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. IEEE. 2020, pp. 77–84.

[117] Dimitrios Spatharakis et al. "A scalable edge computing architecture enabling smart offloading for location based services." In: *Pervasive and Mobile Computing* 67 (2020), p. 101217.

[118] Engin Zeydan et al. "A Multi-criteria Decision Making Approach for Scaling and Placement of Virtual Network Functions." In: *Journal of Network and Systems Management* 30.2 (2022), pp. 1–36.

[119] Linh-An Phan, Taehong Kim, et al. "Traffic-Aware Horizontal Pod Autoscaler in Kubernetes-Based Edge Computing Infrastructure." In: *IEEE Access* 10 (2022), pp. 18966–18977.

[120] László Toka et al. "Machine learning-based scaling management for kubernetes edge clusters." In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 958–972.

[121] Prometheus. https://prometheus.io/. (Last Accessed on 07/01/2023).

[122] Horizontal-Pod-Autoscaling. `https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/`. 2023.

[123] Custom-Pod-Autoscaler. https://custom-pod-autoscaler.readthedocs.io. 2023.

[124] Leila Ismail and Huned Materwala. "Computing Server Power Modeling in a Data Center: Survey, Taxonomy, and Performance Evaluation." In: *ACM Comput. Surv.* 53.3 (2020). ISSN: 0360-0300. DOI: `10.1145/3390605`. URL: `https://doi.org/10.1145/3390605`.

[125] Chaoqiang Jin et al. "A review of power consumption models of servers in data centers." In: *Applied Energy* 265 (2020), p. 114806. ISSN: 0306-2619. DOI: `https://doi.org/10.1016/j.apenergy.2020.114806`.

[126] Ioannis Dimolitsas et al. "AHP4HPA: An AHP-based Autoscaling Framework for Kubernetes Clusters at the Network Edge." In: *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE. 2022, pp. 2566–2571.

[127] Dimitrios Spatharakis et al. "A Lightweight Software Stack for IoT Interoperability within the Computing Continuum." In: *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE. 2023, pp. 715–722.

[128] Ioannis Dimolitsas, Dimitrios Dechouniotis, and Symeon Papavassiliou. "Time-efficient distributed virtual network embedding for round-trip delay minimization." In: *Journal of Network and Computer Applications* (2023), p. 103691. DOI: `10.1016/j.jnca.2023.103691`.

[129] Angelos Pentelas and Panagiotis Papadimitriou. "Service function chain graph transformation for enhanced resource efficiency in NFV." In: *2021 IFIP Networking Conference (IFIP Networking)*. IEEE. 2021, pp. 1–9.

[130] Dimitrios Spatharakis et al. "Distributed Resource Autoscaling in Kubernetes Edge Clusters." In: *2022 18th International Conference on Network and Service Management (CNSM)*. IEEE. 2022, pp. 163–169.

[131] Ioannis Dimolitsas et al. "Multi-Application Hierarchical Autoscaling for Kubernetes Edge Clusters." In: *2023 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE. 2023, pp. 291–296.

[132] George Papathanail et al. "COSMOS: An orchestration framework for smart computation offloading in edge clouds." In: *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2020, pp. 1–6.

[133] Anupama Mampage, Shanika Karunasekera, and Rajkumar Buyya. "A holistic view on resource management in serverless computing environments: Taxonomy and future directions." In: *ACM Computing Surveys (CSUR)* 54.11s (2022), pp. 1–36.