UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

SFC placement and lifecycle management in edge-enabled 5G architectures

Ioannis Sarrigiannis

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

PhD program in Signal Theory and Communications

SFC Placement and Lifecycle Management in Edge-enabled 5G Architectures

Doctoral Thesis by:
Ioannis Sarrigiannis

Supervisors:
Dr. Angelos Antonopoulos
Dr. Konstantinos Ramantas

Tutor:
Dr. Miguel Angel Lagunas Hernandez

Department of Signal Theory and Communications
Polytechnic University of Catalonia

Barcelona, September 2022

Abstract

The fifth generation (5G) of mobile communications urges software defined networking (SDN) and network function virtualization (NFV) to join forces with the multi-access edge computing (MEC) cause. Thus, reduced latency and increased capacity at the edge of the network can be achieved, to satisfy the requirements of a diverse ecosystem where multiple virtualization technologies (VTs) may be employed.

5G service providers highly rely on network softwarization for addressing the latency and resource demanding use cases of the 5G vertical industries, where both applications and network functions are incorporated into virtual network functions (VNFs). If not properly orchestrated, the flexibility of deployment, placement and lifecycle management (LCM) that the VNFs manifest, may cause serious issues in the NFV scheme. As the service level agreements (SLAs) of the 5G applications compete in an environment with traffic variations and VNF placement options with diverse compute and network resources, online placement techniques and advanced LCM approaches become mandatory. The local breakout that MEC-enabled architectures offer, signify that the data will not have to cross the entire network until they reach the application and return to the user, as the application is running at the MEC that is, usually, collocated with the access network.

In this thesis, we tackle the challenges that arise in core and edge-enabled 5G architectures, and we elaborate how the different schemes of VNF deployments as VTs can be used and collaborate to support the automotive and Internet of things (IoT) use cases, as well as more demanding use cases that derive from the combination of various vertical industries. We go beyond the current state-of-the-art solutions by proposing innovative resource allocation and LCM-enabled scheduling algorithms. Furthermore, we focus on the challenges that arise in

single-domain or federated environments, where the flexible placement and LCM of VNFs is supported by the NFV orchestrator (NFVO) that is able to manage, on-the-fly, the network functions and resources. Additionally, we deal with the limitations of the intelligence of the NFVO by providing service aware latency-based embedding and placement mechanisms. Moreover, we propose online scheduling and LCM algorithms where the VNFs are instantiated, scaled, migrated and destroyed based on the actual traffic, respecting the SLA of the 5G use cases, as well as the overall deployment and execution cost.

Our work is supported by our versatile, open-source and custom-built 5G experimental platform, modified and adapted at each architecture to support the corresponding 5G use cases. The experimental results demonstrate the efficiency of our innovative algorithms and their efficacy to deal with the VNF placement and resource reallocation and optimization problems in actual demanding environments with unpredictable traffic schemes. Our methods allow the maximization of the total number of users served and the adaptation to real-time incoming traffic. At the same time, the application SLAs are respected in a cost-efficient way.

Resumen

La quinta generación de comunicaciones móviles (5G) insta a las redes definidas por software (SDN) y la virtualización de funciones de red (NFV) a unir fuerzas con la computacion perimetral multiacceso (MEC). Por lo tanto, se puede lograr una latencia reducida y mayor capacidad en el borde de la red para satisfacer los requisitos de un ecosistema diverso donde se pueden emplear múltiples tecnologías de virtualización (VT).

Los proveedores de servicios 5G dependen en gran medida de la softwarización de la red para abordar los casos de uso de latencia y demanda de recursos de las industrias verticales 5G, donde tanto las aplicaciones como las funciones de red se incorporan a las funciones de red virtual (VNF). Si no se orquesta adecuadamente, la flexibilidad de implementación, ubicación y gestión del ciclo de vida (LCM) que manifiestan los VNFs puede causar problemas graves en el esquema de NFV. A medida que los acuerdos de nivel de servicio (SLA) de las aplicaciones 5G compiten en un entorno con variaciones de tráfico y opciones de ubicación de VNF con diversos recursos informáticos y de red, las técnicas de ubicación en línea y los enfoques LCM avanzados se vuelven obligatorios. La ruptura local que ofrecen las arquitecturas habilitadas para MEC significa que los datos no tendrán que atravesar toda la red hasta que lleguen a la aplicación y regresar al usuario, ya que la aplicación se ejecuta en el MEC, y por lo general, se ubica junto con la red de acceso.

En esta tesis, se abordan los desafíos que surgen en las arquitecturas 5G centrales y habilitadas para el borde, y se elaboran tanto los diferentes esquemas de implementaciones de VNF como VTs se pueden usar y colaborar para respaldar los casos de uso automotriz y de Internet de

las cosas (IoT), así como casos de uso más exigentes que se derivan de la combinación de varias industrias verticales. Vamos más allá de las soluciones de vanguardia actuales al proponer una asignación de recursos innovadora y algoritmos de programación habilitados para LCM. Además, nos enfocamos en los desafíos que surgen en entornos de un solo dominio o federados, donde la ubicación flexible y LCM de VNF está respaldada por el orquestador de NFV (NFVO) que puede administrar, sobre la marcha, las funciones de red y recursos. Además, nos ocupamos de las limitaciones de la inteligencia de la NFVO al proporcionar mecanismos de ubicación e incorporación basados en la latencia y mecanismos conscientes del servicio . Además, proponemos la programación en línea y algoritmos LCM donde las VNFs se instancian, escalan, migran y destruyen en función del tráfico real, respetando el SLA de los casos de uso de 5G, así como el coste general de implementación y ejecución.

Nuestro trabajo está respaldado por nuestra plataforma experimental 5G versátil, de código abierto y personalizado, modificada y adaptada en cada arquitectura para admitir los casos de uso 5G correspondientes. Los resultados experimentales demuestran la eficiencia de nuestros algoritmos innovadores y su eficacia para hacer frente a los problemas de ubicación y reasignación de recursos y optimización de VNF en entornos exigentes reales con esquemas de tráfico impredecibles. Nuestros métodos permiten la maximización del número total de usuarios atendidos y la adaptación al tráfico entrante en tiempo real. Al mismo tiempo, los SLAs de la aplicación se respetan de forma rentable.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The exponential increase on requests for a variety of diverse services creates the need for an omnipresent network, which should be faster, more responsive and reliable, and easily accessed under any conditions, characteristics that the current network technology cannot support [1]. The fourth generation long term evolution (4G LTE) network technology pioneered the 2010s, improving successfully the previous mobile network generation, the third generation (3G), mainly in the high-speed mobile broadband access part. In the recent years, LTE becomes overloaded, struggling or, in cases, failing to cope with current and future service needs.

The successor of the LTE network technology is the fifth generation (5G) [2]. According to the 3rd generation partnership project (3GPP), the 5G use cases can be split into three different categories. The ultra-reliable low-latency communication (URLLC) requests that have a strict latency requirement, the enhanced mobile broadband (eMBB) that services demand high bandwidth Internet access, and the massive machine type communication (mMTC) applications that involve connectivity of a large number of devices that transmit periodically a low amount of traffic.

The different 5G use cases (i.e., services provided by 5G vertical industries, or 5G verticals) and applications belong to at least one of the aforementioned categories [2]. They include, among others, the automotive vertical, which includes autonomous driving or navigation and traffic jam avoidance services, the Internet of things (IoT) use cases, which include energy, agriculture and city management services, and the media entertainment vertical, which includes ultra high definition (UHD) streaming or broadcasting and augmented reality services. For instance, automotive applications may require strict latency for autonomous driving, handover processes to cover their mobility needs, or high compute resources during rush hours. IoT use cases may need to support an increased amount of interconnected devices that, when aggregated, result in a high throughput or compute resources requirement, with latency not being that strict. Finally, combinations of some verticals, such as automotive and entertainment for augmented reality and autonomous driving and navigation, may lead to increased needs in terms of throughput, coverage, compute resources, and latency requirements.

With the adoption of the software defined networking (SDN) technology [3], combined with the network function virtualization (NFV) concept [4], the legacy and monolithic network infrastructure transforms into a diversified and adaptive solution, enabling the underlying layer to meet the fast, vast and seamless communication needs of the new era. Hence, the network consists of virtual network functions (VNFs), which are managed by the NFV orchestrator (NFVO) that enables lifecycle management (LCM) techniques for additional control and interoperability [5]. The network slicing approach [6] introduces the required isolation, service flexibility and automation, as it introduces the required network and resource allocation per VNF, over the same network infrastructure. In terms of virtualization technology (VT), the VNFs can be implemented as virtual machines (VMs)[1], containers[2], or unikernels[3]. Benefiting from the VT concept, the software applications that the 5G verticals support can be provided as application VNFs. This virtualized way enables them to be deployed, orchestrated and managed efficiently, in accordance with the NFV paradigm, considering the actual needs of the end users.

The restrictions of the location where VNFs can be deployed have been loosened. From a centralized (or core) approach, where the cloud computing paradigm prevails, to a decentralized (or edge) one, where multi-access edge computing (MEC) [7] techniques dominate, VNFs can be positioned from the central datacenter, up to few meters from the user equipment (UE), respectively. Additionally, in some cases the VNFs can be deployed across the whole network, even at the net-

[1]VMs are virtual environments that function as virtual computer systems, having allocated to them their own physical resources.

[2]Containers are lightweight packages of software that contain all of the necessary elements to run in any environment.

[3]Unikernels are unique, single process systems that run in a single address space.

work edge, in accordance with the fog computing paradigm [8]. The positioning of the VNFs across the network has a great impact on the quality of service (QoS) that is defined by the service level agreement (SLA), a pact between the customer and the provider. The decision where the VNFs will be positioned across the path from the central location to the UE proximity is called the VNF placement problem. When multiple VNFs are connected to provide one service, they create VNF forwarding graphs (VNFFGs) or service function chains (SFCs)[4].

The telecommunications industry is constantly focusing on decreasing the end-to-end (E2E) latency, in order to mainly support the latency demanding 5G verticals. The local breakout that MEC-enabled architectures offer, signify that the data will not have to cross the entire network until they reach the application and return to the user, as the application is running at MEC hosts that are, usually, collocated with the access network. Therefore, the high latency and the increased backhaul channel[5] utilization that are observed when cloud resources are used, are greatly reduced. Despite being an attractive concept, certain aspects need to be further investigated. Without loss of generality, MEC compute resources are less powerful than the cloud counterparts [9]. MEC devices may be spread in a large geographic area with limited or restricted access, making their installation, upgrade or repair more time-consuming and costly, compared with the centralized datacenters where access and maintenance are faster and more cost effective, respectively. Therefore, it is imperative to efficiently optimize the allocation of the MEC resources, given their unique attributes and higher cost.

The concept of fog computing further extends the edge computing paradigm. Fog computing "distributes computing, storage, control and networking functions closer to the users, along a cloud-to-thing continuum"[10]. Therefore, VNFs can be deployed and executed across the whole path along the core network and the UE. Without limiting the generality of the foregoing, VNFs may be located in generic purpose servers, routers, switches, vehicles or smartphones, known as fog nodes. Prior to the fog computing paradigm, these devices were limited to operate autonomously; now they can participate in an interconnected environment that can benefit from the unified resources. Consequently, the ability to leverage the available resources, particularly at the UE level, can guarantee the ultra-low latency

[4]In this thesis we interchangeably use the the terms VNFFG and SFC.

[5]Backhaul channel is the channel that connects an access node with the core network.

that various 5G verticals require. However, the complexity is increased, as the fog nodes consist of different underlying hardware architectures with diverse network and compute resources, and explicit VT support. These limitations need to be taken into consideration when designing applications that can be executed in such heterogeneous environments.

In the economic aspect, a 3 bn euros in capital expenditure (CAPEX) investment is anticipated in the European Union (EU) by 2030, only for the automotive vertical [11]. Additionally, for the entertainment vertical the estimation expects a 730 mil euros CAPEX investment by 2030, while the operational expenditure (OPEX) cost forecast reaches the amount of 214 million euros [12]. Taking into consideration the high acquisition and operational costs, mobile network operators (MNOs), as the main providers of MEC, foresee the need of collaborations, in order to decrease their CAPEX and OPEX costs. Only from the automotive vertical, 275 mil euros can be saved up by a synergy of MNOs. In the case of the entertainment vertical, covering just 20% of the population in rural areas has the same running cost as half of the running cost of the network of the entire country. These are just two examples of how a cross-operator collaboration benefits the CAPEX and OPEX cost of the MNOs. The local MNO can expand its network and compute area of coverage by using, on-demand, the core or edge resources of a foreign MNO, in a resource federation scheme[6].

Towards enabling the cross-operator collaboration, the global system for mobile communications association (GSMA) introduced the operator platform concept (OPC), where operators collaborate to offer a unified edge platform [13]. In its first phase, the OPC will federate multiple edge computing infrastructures of multiple MNOs. Thus, application providers will be granted access to a global edge cloud, making it possible to run their services nationwide, without being restricted by the limitations of each individual MNO. This opportunity opens the window to support more resource and latency demanding use cases, such as augmented and virtual reality applications, where the placement of the application may have an impact in the QoS and the operational cost.

Considering the diversity of the underlying infrastructure, in terms of compute and network resource availability and ownership, the VNFs must be placed in such

[6]The local MNO has a collaboration agreement with at least one foreign MNO, under which the former may use the network and compute resources of the latter, on-demand and in a pay-as-you-go approach.

locations where the SLAs can be respected. Additionally, the optimal positioning in terms of cost and profitability, resource allocation and high service availability, are some aspects that need to be further investigated. It is of paramount importance for a 5G-enabled network to i) be constantly adapting to the resource demands of the applications it serves, ii) consider the continually changing needs of the UEs, including but not limited to their mobility, iii) understand and adapt to the unpredictable traffic patterns that may occur, and iv) be resilient by handling possible infrastructure failures [14]. The aforementioned challenges need to be taken seriously under the consideration of any VNF placement tactic that may be employed.

Currently, offline [15–17] and online [18–20] VNF placement approaches can be found in the literature. On the one hand, in the offline placement method, the placement decision is taken a priori in order to satisfy a finite and known number of UE requests, with a given request lifetime, respecting certain resource constraints. On the other hand, the VNF placement decision in the online approach is taken in real-time when the UE request arrives. It takes into consideration real traffic data and current resource constraints upon receiving the UE request to further enhance the placement decision. While the online approach is more efficient and realistic upon accepting a new service, it still lacks the supervision of the underlying compute and network resources during the full lifecycle of the said service.

Throughout the lifecycle of a VNF, the load in terms of compute resources might temporarily increase (e.g., due to increased requests by one or more UEs that are requesting the services of that VNF), and vice versa. Scaling is the LCM action that can manage such load changes and provide the resources needed to cover the increased traffic, or release the resources when the load restores back to normal conditions. In addition, VNFs might need to change the host environment between core and edge resources, in order to maintain the QoS of a UE (e.g., when a handover process takes place due to mobility). Therefore, migration LCM actions facilitate such location adaptations. Consequently, LCM and orchestration techniques, such as scaling and migration, need to be engaged to further handle the traffic pattern changes that may occur.

The aforementioned combination of VNF placement methods with the LCM techniques should also consider federated architectures, where multiple MNOs cooperate in order to support 5G vertical applications, expanding their geographic coverage in a cost-effective manner. In the literature, there are very few works [21–23] that tackle VNF placement, orchestration or LCM techniques in NFV-based multi-domain environments. Additionally, the proposed algorithms should not only be required to be tested in simulation-based settings, but also in a controlled real-like environment. Therefore, applications can be tested and executed under various realistic parameters and conditions, extracting results that could be proven beneficial for the actual 5G implementations.

Given the restrictions of the MEC resources (e.g., restricted resources, higher cost, etc.), not all VNFs can or should be placed at the UE proximity. Depending on the SLA of each application, VNFs might need to be placed near the UE proximity (e.g., latency-critical applications), or it might be equally acceptable, in terms of QoS, to be placed at the core (e.g., latency-tolerant applications). Therefore, novel VNF placement methods that will take into consideration the particularities of each system, application and UE, are needed. The location (i.e., the core or the edge resources) where to deploy a VNF, or each VNF of an SFC, can have an impact on the cost, the smooth operation of the application or the co-hosted applications, and the user experience. Thus, a thorough study that takes into consideration the aforementioned parameters is needed.

To that end, in this thesis, we attempt to fill the gap in literature regarding the combined study of the SFC placement methods in different 5G architectures and the adoption of the LCM techniques, tested in small-scale open-source based implementations that express the same behaviour with the actual real-life 5G environments. Motivated by the diverse, latency and resource demanding 5G use cases, (e.g., the automotive vertical for the low latency requirements, and the entertainment vertical for the high demand in compute resources), we employ three different 5G architectures; a fog-enabled single-domain architecture to support the automotive vertical, a MEC-enabled single-domain architecture to support the IoT use cases, and a federated MEC-enabled architecture to support the more demanding applications that derive from the combination of the vertical industries.

Moreover, we set the NFV-enabled environment that employs the VT tools as means for delivering the 5G applications as VNFs, or SFCs, to the UEs. Then, we propose different online SFC placement algorithms that apply to the different architectures, in order to facilitate the initial placement of the VNFs, while, we propose LCM-enabled algorithms for the smooth management of the entire lifecycle of the VNFs, enhancing, thus the lifecycle visibility that SFC placement algorithms lack of, demonstrating, at the same time, the beneficial impact on the interplay of centralized, edge or foreign resources. Following the network slicing and orchestration principles, the common underlying infrastructure is shared, where the needed resources are allocated to each VNF, without losing the required isolation. Therefore, we guarantee that the VNFs have access to the required resources upon request, while the resources are released to the system when they are not required anymore. Finally, we deploy a 5G experimental platform and we adapt it to each use case and environment, gaining valuable insights from the experiments.

The aim of this thesis is to provide the tools to support the diverse 5G use cases in different environments and under distinct occasions. Our goal is to i) respect the 5G use case SLAs, ii) maximize the UE request acceptance (or, respectively, to decrease the service block rate), and iii) maximize the profit for the MNOs, overcoming the unstable traffic patterns, the UE mobility, the limited resources, their scarcity and their cost diversity. This study was performed in order to accelerate the transitioning from the monolithic network architecture, where services were running only centrally, to a service-based modular framework, where applications from different providers can be executed across the whole network.

As experimentally demonstrated, MNOs can have significant benefits from this thesis. In more detail, by applying our suggested LCM-based intelligence to the fog-enabled architecture, a 183% increase can be observed in the accepted latency-critical incoming requests, without any SLA violation. By adopting our proposed VNF placement and orchestration algorithms in the MEC-enabled single-domain architecture, a 50% increase of accepted latency-critical incoming requests can be achieved when both edge and core resources are employed, compared to only edge ones. Additionally, by applying our cost-based SFC placement and LCM-based algorithms in a federated architecture, MNOs can work towards the aforementioned 275 mil euros savings in CAPEX and OPEX costs for the automotive vertical. The cost-related benefits, though, can also extend to other verticals, as our work is able to support all types of 5G use cases. Simultaneously, MNOs can increase their resource capacity and coverage presence by using, on demand, the infrastructure of a collaborative operator, without additional CAPEX investments, which can lead to an up to 54% decrease in service block rate, compared to non-federated approaches.

The structure of the thesis and the main contributions of our work are discussed in detail in the following section.

1.2 Structure of the thesis and contributions

As explained in the previous section, the motivation of this thesis has stemmed from the need to investigate at full extend new VNF placement algorithms, combined with LCM techniques that are applied to diverse network architectures that combine the centralized with decentralised resources, in real-like 5G environments. Therefore, in this thesis we propose online VNF placement algorithms, migration and scaling LCM techniques that get activated in distinct occasions, which are applied over different architectures, where multiple VTs are employed.

The remaining part of the thesis consists of five chapters. Chapter 2 provides some necessary background information concerning the tools and technologies we consider for our solutions; the SDN, the NFV, the NFVO, the 5G core network functions (5GC NFs), the edge technologies, the VTs, and the key enabler open-source and commercial tools. The innovative contributions of the thesis are organized into three chapters. In Chapter 3, where a fog-enabled architecture is employed, we demonstrate how the use cases of the automotive vertical can be benefited from the parallel use of VMs, containers and unikernels as SFCs than can be deployed from centralized locations, up to the UE proximity, where the scaling and migration decisions are taken to tackle different real-life issues. In Chapter 4, where a MEC-enabled 5G IoT architecture is represented, we introduce our first VNF embedding algorithm as well as our online VNF scale-out/scale-in and dynamic live-migration scheduling algorithm. In Chapter 5, where a multi-domain MEC-enabled 5G architecture is employed, we focus on optimizing the SFC placement and execution cost of an operator and service provider, by proposing our hop-based heuristic SFC placement algorithm, as the most effective solution compared to baseline and existing state-of-the-art (SoA) approaches. Finally, Chapter 6 discusses the conclusions of the presented thesis and identifies potential lines for future investigation. In the following, the main contributions of the thesis will be outlined in more detail.

In Chapter 3, we focus on a fog-enabled architecture that utilizes the whole path from the central resources to the resources at the UE level to provide applications for the automotive vertical. The applications are represented as connected VNFs, thus constitute SFCs, while the VTs that implement them is a combination of VMs, containers and unikernels. In this chapter, we introduce the scalability and migration LCM techniques, as a means to handle unexpected increased traffic. Additionally, the migration LCM technique is also employed as a way to tackle the handover process that takes place due to the UE mobility, as well as to bring the service to the closer to the user, when traffic demands increase. The usefulness of both LCM techniques is demonstrated, as the custom built 5G experimental platform is used to provide experimental results that increase the requests that can be handled by the system.

In Chapter 4, we investigate the online VNF placement methods, enhanced with LCM techniques in a MEC-enabled IoT architecture. The 5G cloud applications are implemented in the form of VNFFGs that result in VNF chaining. We categorize the VNFs based on their delay constraints and their priority as high or low priority latency-critical VNFs (HP/LP LCVNFs) and latency-tolerant VNFs (LTVNFs). We introduce a VNFFG embedding algorithm of virtualized chained services, taking into account their latency requirements and service priorities. Then, we propose an online VNF scaling and dynamic live migration scheduling algorithm for the real-time allocation of the VNFs to the MEC and cloud resources, leveraging real-time service LCM features to meet the UE requests. A 5G experimental platform is implemented and various experiments demonstrate that with the proposed schemes a better utilization of MEC and cloud resources can be obtained on-the-fly, enabling the system to serve a higher number of latency-critical applications, without SLA violation.

Finally, in Chapter 5, we investigate a cost-aware placement and enhanced LCM of SFCs in a multi-domain 5G architecture. In this scenario, a local provider can expand its compute and network resources by using resources from a collaborative foreign provider. The cross-operator collaboration is further supported by an online integer linear programming (ILP) formulation, constructing the SFC placement problem as a ILP optimization problem, and a less complex online hop-based heuristic algorithm that finds a near-optimal solution for the SFC placement problem. Both approaches are cost-aware, taking into consideration the latency network constraint. Additionally, a live migration algorithm is proposed for the migration of the SFCs that were placed in the foreign infrastructure, when local resources permit it. When applying the algorithms and LCM techniques to our 5G experimental platform, we demonstrate the benefits in terms of complexity, service block rate, SFC execution cost and increased traffic handling.

1.3 Research Contributions

The work presented in this thesis, has been published in three journals, in one international conference and one book chapter. The list of publications is as follows:

[J3] **I. Sarrigiannis**, A. Antonopoulos, K. Ramantas, M. Efthymiopoulou, L. M. Contreras, C. Verikoukis, *"Cost-aware placement and enhanced lifecycle management of service function chains in a multi-domain 5G architecture", IEEE Transactions on Network and Service Management*, 2022.

[J2] **I. Sarrigiannis**, Luis M. Contreras, K. Ramantas, A. Antonopoulos and C. Verikoukis, *"Fog-enabled Scalable C-V2X Architecture for Distributed 5G and Beyond Applications ", IEEE Network*, vol. 34, no. 5, pp. 120–126, 2020.

[J1] **I. Sarrigiannis**, K. Ramantas, E. Kartsakli, P-V. Mekikis, A. Antonopoulos, C. Verikoukis ," Online VNF Lifecycle Management in a MEC-enabled 5G IoT Architecture", *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4183–4194, 2020.

[C1] **I. Sarrigiannis**, E. Kartsakli, K. Ramantas, A. Antonopoulos, C. Verikoukis et al., *"Application and Network VNF migration in a MEC-enabled 5G Architecture," in 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Sep. 2018, pp. 1–6

[BC1] A. Okic, **I. Sarrigiannis**, U. Fattore, B. Xiang, A. E. C. Redondi, E. Di Nitto, A. Antonopoulos, K. Ramantas, C. Verikoukis, L. M. Contreras, M. Liebsch, *"Resource Management for Cost-Effective Cloud Services", in book Enabling 6G Mobile Networks*, pp 399-435, Springer, Cham, January 2022.

Chapter 2

Background

2.1 Introduction

The main goal of this thesis is to provide tools towards the support of diverse 5G use cases in different environments, under distinct occasions. Consequently, a wide variety of methods, cutting edge technology tools and key enabler technologies need to be combined together. Additionally, in conjunction with our proposed SFC placement and LCM-enabled algorithms, our work will be able to provide the 5G use cases with the required low latency and decreased service block rate, while maximizing the profit of the MNOs by using our proposed CAPEX and OPEX cost-effective tactics.

To that end, in this chapter, we will provide the background behind our system models that will facilitate the understanding of the contributions of this thesis. Hence, Section 2.2 discusses the SDN, which is one of the 5G key enablers that enables the separation of the network software from the proprietary hardware, making it possible for the NFs to run closer to the UE. Similarly, Section 2.3 provides the NFV reference architecture, based on which the physical network functions (PNFs) become virtual, therefore eliminating the restrictions that the legacy technology imposes. Furthermore, Section 2.4 explains in detail the role and the actions of the NFVO, the coordinator of the compute and network resources needed to set up the VNFs. Additionally, in Section 2.5, the 5GC NFs are explained, as well as their mapping to the NFs of the 4G technology. Moreover, the various edge architectures are presented in Section 2.6, while the network slicing technology, responsible for the allocation of the network and compute resources to slices of services, is explained in Section 2.7. Finally, Section 2.8 discusses and compares the three VTs, as key enablers for the virtualization environments that the network software and applications run, while 2.9 presents some fundamental open-source and proprietary virtualization and NFVO tools.

2.2 Software defined networking

Telecommunication networks are nowadays built using specific proprietary equipment to facilitate a specific need of one or more network functions. Typical equipment, such as routers, switches, base stations and voice gateways, is monolithic because the devices consist of a certain hardware, software and associated management systems [24]. SDN is an emerging networking paradigm that attempts to eliminate the limitations that the traditional vertical networks impose. By proposing the separation of the control from the data plane, the control logic can be performed by a centralized platform while the network switches become forwarding nodes. For the separation to be achieved, a well-defined programming interface is crucial between the switches and the SDN controller, utilizing application programming interfaces (APIs).

As a result, the proprietary hardware and embedded software that were previously used have been replaced by software that can be deployed, executed, maintained and updated on standard hardware and servers where the decision making is hosted (i.e., control plane) and the decisions are sent to the network equipment (i.e., switches, routers) that simply execute and propagate (i.e., data plane) the decisions across the network. The architectural components include:

- the SDN applications, which are programs that communicate their network requirements via a northbound interface,
- the SDN controller, which is the centralized entity responsible for translating the requirements from the upper level to the data paths, as well as for providing the SDN application an abstract overview of the network, and
- the SDN data-path or network infrastructure, which is the logical network device, that could be part of one or more physical networks, responsible for the visibility and the control of the network flows, communicating with the SDN controller through a universal and vendor neutral southbound API.

2.3 Network function virtualization

Legacy PNFs require a proprietary hardware in order to be deployed and executed. NFV technologies add new capabilities to the current telecommunication networks by allowing a variety of innovative operations to be included to the deployment process, thus achieving significant alleviation of the constraints the legacy network technologies impose. Moreover, with the aid of management and orchestration (MANO) modules, NFV demonstrates its potential to transform and evolve the traditional network management process from PNFs to native NFV management, with an extreme high grade of automation [25]. With the NFV we are able to virtualize the SDN components into VNFs that are deployed as VMs and speed up the initial deployment time. Finally, based on live traffic patterns or business models, MNOs can dynamically place and reallocate the VNFs across datacenters and edge locations, overcoming, thus, the limitations that the PNFs impose.

The Day-0 actions conventionally include the physical network functions installation and the initial configuration to make them accessible and reachable by setting up users, credentials, networks etc. Furthermore, Day-1 processes introduce operations such as license activation and network and neighbor configuration, while Day-2 operations include service and business provisioning. Correspondingly, MANO substitutes those processes with Day-0 VNF and network service deployment, as the traditional physical resources have been replaced with virtual ones. Additionally, VNF configuration, as well as license activation and neighbor configuration, are part of the procedures of both Day-0 and 1, providing a subtle border between those 2 days, in contrast with the hard-specified actions that the traditional deployment method dictates. Finally, service and business provisioning are part of Day-2 configurations, again by managing the already deployed virtual and not physical functions.

As an evolution from VNFs, cloud-native network functions (CNFs) are designed and implemented to run inside containers. This containerization of the network functions further allows the rearchitecting of the network function software by introducing microservices, improving thus the flexibility and agility of the system. During the transitioning period from the PNFs to the VNFs and CNFs, all three type of network functions are expected to seamlessly coexist and interact.

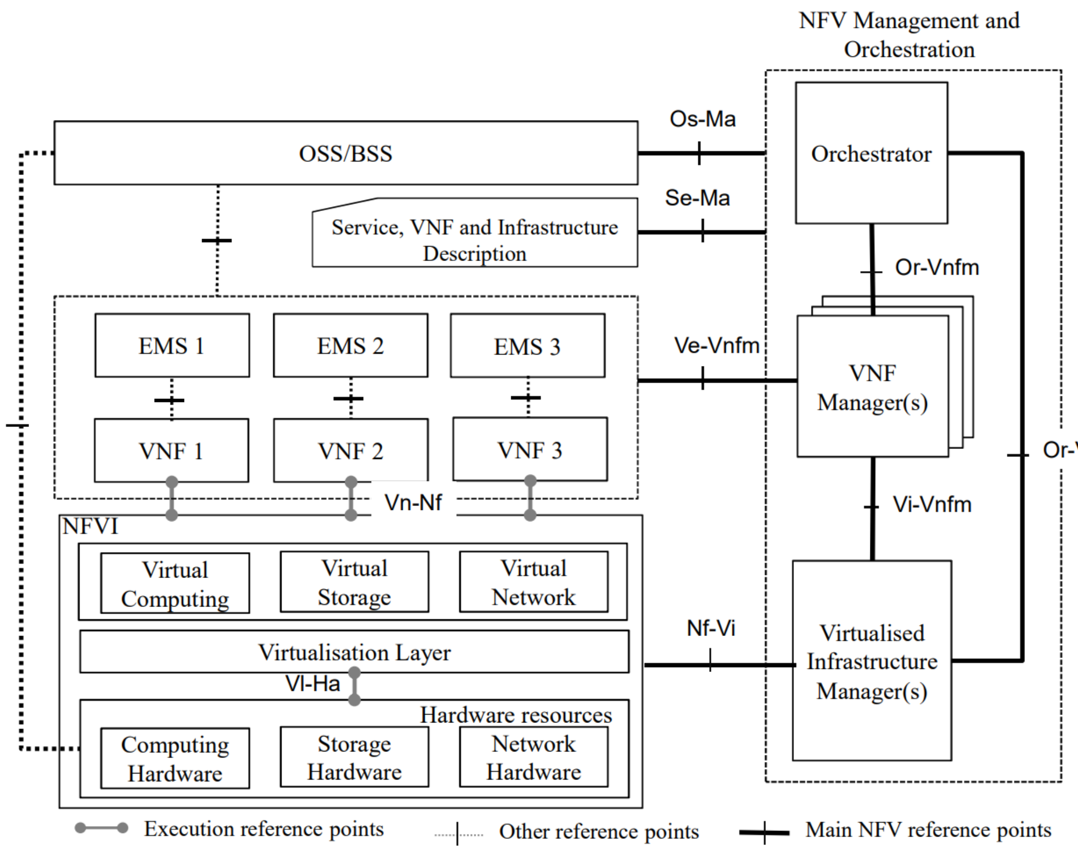

Figure 2.1: ETSI NFV reference architectural framework

The European telecommunications standards institute (ETSI) NFV architectural framework [26] focuses on the changes that will occur in an operator's network because of the NFV process. Figure 2.1 [26] describes the functional blocks:

- Business support system (BSS): it is a computer system used by a telecommunication company to run the customer-facing business operations.
- Operations support system (OSS): it is a computer system used by a telecommunication company to manage its networks.
- VNF: it is a virtualization of the network function in a legacy non-virtualized network. It can be composed of one or multiple internal components (VMs or containers).
- Element management system (EMS): Performs the typical management functionality for one or several VNFs.

- NFV infrastructure (NFVI): The hardware and software components that create the ecosystem on which VNFs are deployed, managed and executed.
- NFVO: the component in charge of the orchestration and management of the NFV infrastructure and software resources, and realizing network services on NFV Infrastructure.
- VNF manager (VNFM): the component responsible for VNF lifecycle management.
- Virtualized infrastructure manager (VIM): comprises the functionalities that are used to control and manage the interaction of a VNF with the physical resources (i.e., computing, storage, networking), as well as their virtualization.

The NFVO is the coordinator of the network and compute resources that are allocated to the VNFs. The NFVO is an important component for our work, as it enables the management of the VNFs. Therefore, we further explain it in the next section.

2.4 NFV orchestrator

The NFVO, a key component of the NFV MANO architectural framework, constitutes the central controller of the system. It is responsible for the key SFC LCM operations, such as the i) instantiation, ii) scaling, iii) migration, iv) update/upgrade, and v) termination of the VNFs that form the SFC. While the NFVO is the central brain of the system, the VIM is the module that actually executes the operations.

- The instantiation refers to the creation of the VNF, using on-boarding artifacts in order to allocate the required resources. It includes Day-0, Day-1 and Day-2 configuration that include instantiation and management setup, services initialization and runtime operations of the VNFs, respectively.
- The scaling refers to the modification of the pre-allocated resources of VNFs, including, but not limited to, central processing unit (CPU), random access memory (RAM) and storage. The two scaling methods include the vertical scaling, where the resources of a VNF can be increased (i.e., scale-up) or decreased (i.e., scale-down), and the horizontal scaling, where more copies

of the original VNF are created (i.e., scale-out) or destroyed (i.e., scale-in). The main advantage of the horizontal scaling is that this technique does not require any VNF downtime, compared with the vertical scaling that does require the VNF to be stopped for the reallocation of the resources to be completed, and restarted thereafter.

- The migration refers to the moving of a VNF to a different physical machine. The two migration methods include the non-live migration, where the VNF is shut down for a period of time in order to be moved, and the live migration, where the moving process takes place without service interruption [27].

- VNF update and upgrade refer to the support of VNF software and the configuration changes that may need to occur during the life of the VNF. The VNF update is the ability to execute an application software modification, while the VNF upgrade might introduce new functionalities or interfaces that are needed to maintain the service continuity and availability.

- Finally, the VNF termination refers to the last step of the lifecycle of a VNF that occurs when the VNF has completed its purpose. At this step, the VNF is destroyed and the resources are released to the resource pool of the system, ready to be re-used by current or future incoming requests.

2.5 5G core network functions

The 5GC network is the 3GPP standardized core that optimizes the management of control and user plane, shifting from the legacy 4G evolved packet core (EPC) [28]. The main benefits of the new virtualized core include the support for MEC, the native support of network slicing, and the QoS enhancements that lead to better service flexibility, security and automation. In the RAN part, the 5G new radio (NR) facilitates significant improvements compared with 4G LTE, in terms of latency, data rate speed, mobility support, user density and frequency ranges, as shown in Table 2.1.

The legacy 4G EPC NFs use proprietary hardware and software. They mainly consist of the mobility management entity (MME) that is responsible for the UE requests, the home subscriber service (HSS) that is a master user database, the serving gateway (SGW) that routes and forwards user data packets, and the packet

Table 2.1: Comparison between 5G NR and 4G LTE

	Latency	**Data Rate**	**Mobility Support**	**User Density**	**Frequency Range**
5G NR	∼1 ms	peak 20 Gb/s	500 Km/h	∼1000K/km^2	up to ∼52GHz
4G LTE	10-50 ms	peak 300 Mb/s	350 Km/h	∼2K/km^2	up to ∼6GHz

data network gateway (PGW) that connects the EPC to external IP networks. On the other hand, 5G NFs are virtualized and run on a cloud infrastructure. The most important 5GC NFs, along with their main functionalities, are:

- **Network slice selection function (NSSF):** selects the network slice instance. It is a new NF.
- **Network exposure function (NEF):** performs exposure capabilities and events to other NFs or 3rd party applications. It is a new NF.
- **Network repository function (NRF):** supports the discovery of the other NFs. It is a new NF.
- **Policy control function (PCF):** enables the policy rules for governing the network behaviour by providing the rules to other NFs in order to enforce them. It is the evolution of the policy and charging rules function (PCRF).
- **Unified data management (UDM):** handles the user identification, subscription and roaming. It handles some of the functions of the HSS.
- **Application function (AF):** exposes the application layer for interacting with network resources and is responsible for the traffic steering rules creation. Its functionalities resemble with the ones of the AF of the 4G network.
- **Authentication server function (AUSF):** performs the authentication of the subscribers. It handles some of the functionalities of the HSS.
- **Access and mobility management function (AMF):** is responsible for the management of the registration, the connection, the mobility, as well as the authorization and the authentication access of the UEs. Its functionalities resemble with the ones from the MME.

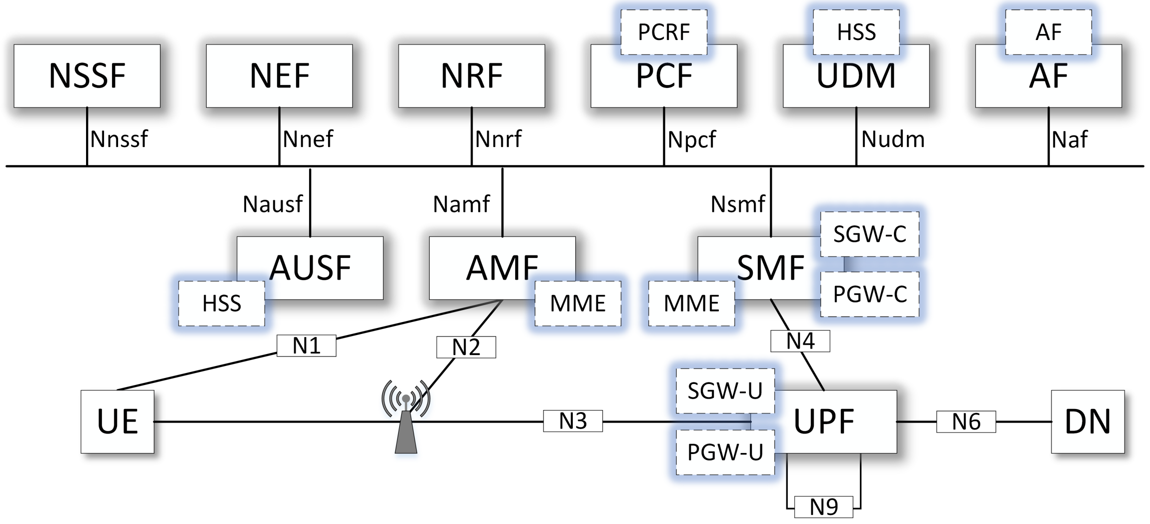

Figure 2.2: 4G EPC NFs mapped to 5GC NFs

- **Session manager function (SMF):** performs the session establishment and the selection of the control and user plane and manages the protocol data unit (PDU) sessions [1]. It handles some of the functionalities that were previously handled by the MME, as well as the SGW and PGW actions that are relevant to the core plane.

- **User plane function (UPF):** enforces the traffic and policy rules, handles the QoS and is responsible for the packet routing and forwarding towards the data network (DN), or towards other UPFs, through the PDU session anchoring. It handles some of the user plane related functionalities that were handled by the SGW and the PGW.

The 5GC utilizes a service based architecture (SBA) where the NRF provides discovery between the NFs, based on subscribe-notify or request-response actions, and complements the point-to-point interface model. Additionally, the separation of the control and user plane (CUPS) will enable independent scaling and flexible deployments on each plane. Finally, 5G NFs are designed to be stateless[2], a key element that enhances their scalability and redundancy.

[1]PDU session is a logical connection between the UE and the data network that provides the application.

[2]5G NFs are not maintaining any states or user data locally, they are only responsible for their functionalities.

The most recent data state that 200 global network operators in 78 countries are offering 5G mobile services [29]. The majority of the MNOs are opting for the transition from the 4G EPC to a 5G non-standalone core (5G NSA), where, usually, the 4G LTE RAN is upgraded to the 5G NR, while the 4G EPC becomes virtualized (vEPC) [30]. In this scenario, each of the virtualized PGW and SGW are split into two functions, the user (vPGW-U and vSGW-U) and the core (vPGW-C and vSGW-C), respectively. The direct use of 5G NR RAN and standalone 5G core (5G SA) is currently supported just by 20 operators in 16 countries [29].

2.6 Edge technologies

Edge technologies [31] extend the cloud computing technologies to the edge of the network. Mobile edge computing or, as renamed in 2017, multi-access edge computing (MEC) was introduced by the ETSI industry specification group (ISG) as a way of pushing the intelligence, as well as high processing and storage capabilities, to the end of the network [32]. The main focus of MEC is to improve the overall network performance by minimizing the network congestion, while optimizing the resource allocation by wrapping SDN and NFV in a well-defined package, in the edge of the network. The most significant benefit of MEC technology is that it provides guaranteed low latency services, towards the realization of the URLLC applications, as it enables the concept of computational offloading (i.e., the transfer of resource intensive computational tasks to an external platform) from the core to the edge. Therefore, MEC facilitates the execution of compute demanding 5G use cases that require ultra-low latency, a task that a centralized cloud solution cannot accomplish. Finally, MEC nodes can act as aggregators by collecting the data from the devices before they reach the core network.

Apart from MEC, edge technologies include fog computing, introduced by Cisco in 2011 [33] and Cloudlets, presented by Carnegie Mellon University in 2013 [34]. The objective is still the same, to bring cloud computing capabilities closer to the end user, but each technology has a unique position to both existing and future networks. Table 2.2 [31] compares in detail all three edge computing implementations.

Fog computing presents a computing layer utilizing heterogeneous devices like machine-to-machine gateways, wireless routers, switches, and vehicles, which are

Table 2.2: Comparison of edge computing implementations

	Fog	**MEC**	**Cloudlet**
Node devices	Routers, Switches, Access Points, Gateway	Servers running in base stations	Datacenter in a box
Node location	Varying between End Devices and Cloud	Radio Network Controller / Macro Base Station	Local / Outdoor installation
Software architecture	Fog Abstraction Layer based	Mobile Orchestrator based	Cloudlet Agent based
Context awareness	Medium	High	Low
Proximity	One or Multiple Hops	One Hop	One Hop
Access mechanisms	Bluetooth, Wi-Fi, Mobile Networks	Wi-fi, Mobile Networks	Wi-Fi
Internode communication	Supported	Partial	Partial

called fog computing nodes, or fog nodes, in order to process and store data from the end users before forwarding them to the cloud or to the MEC. Those devices can be placed anywhere between the UE and the cloud but are exposed to the former as a uniform Fog abstraction layer that provides a set of functions to perform the resources' E2E management and allocation. While MEC can act as aggregator as well, the main difference is that MEC servers are placed within the base station premises, providing the Radio Access Network (RAN) with one-hop cloud computing capabilities. Finally, cloudlets, also referred as datacenter-in-a-box, consist of dedicated devices that bring datacenter capabilities, on a lower scale, to the proximity of the end user. Cloudlets enable computing offloading by provisioning VMs to allocate the dedicated resources the end users request and are provided at a one-hop distance, utilizing wireless LAN networks.

Taking a closer look into the basic network functions for MEC deployment, we distinguish the UPF that is responsible for steering the traffic from the lower to the higher tier, the AMF that is in control of mobility related procedures and the SMF that is in charge of controlling and configuring the UPF for the traffic steering. While logically it is to some extent significant, physically both the placement of the network functions and the MEC are extremely important. Within the diversity of the edge environment, ETSI gives four different example scenarios for physical deployment of the MEC nodes [7], demonstrating the flexibility of the deployment locations. Based on operational, performance or security related requirements, mobile operators are at liberty to select the appropriate location that serves their needs.

2.7 Network slicing

Network slicing [6, 35, 36] is the distinctive 5G technology that makes it possible to support diverse requirements over the same network and compute infrastructure, providing, at the same time, the needed isolation. A network slice is a set of network functions that logically create a virtual network in order to facilitate the needs of specific services. With the virtualization of the network functions it is easy to decompose the monolithic legacy network infrastructure into numerous modular network capabilities, scattered across a cloud infrastructure, the cornerstone of 5G. The management of the VNFs, utilizing machine learning, data analysis and autonomic network management functions will lead to a self-optimized and fully automated network, able to provide a diverse type of services. The various network slices that span across all network tiers, from the UE to the core, raise the need of abstracting the individual network segments into a cohesive E2E vision.

Specific network instances (i.e., slices) can be used for the realization of the three fundamental 5G use cases, namely the URLLC, the eMBB and the mMTC. Despite using the same underlying infrastructure, each slice will be able to provide the guaranteed processing power, storage, and bandwidth for the 5G use cases, respecting, at the same time, the required security and isolation. Moreover, network and compute resources across the network can be allocated either in a dedicated or shared scheme, depending on the SLA of each slice. A dedicated scheme offers the exclusive allocation of the underlying resources, whether the slice is constantly using them or not, while a shared scheme offers the underlying resources to be shared by multiple slices, under the assumption that not all slices will demand the full capacity of the underlying resources simultaneously.

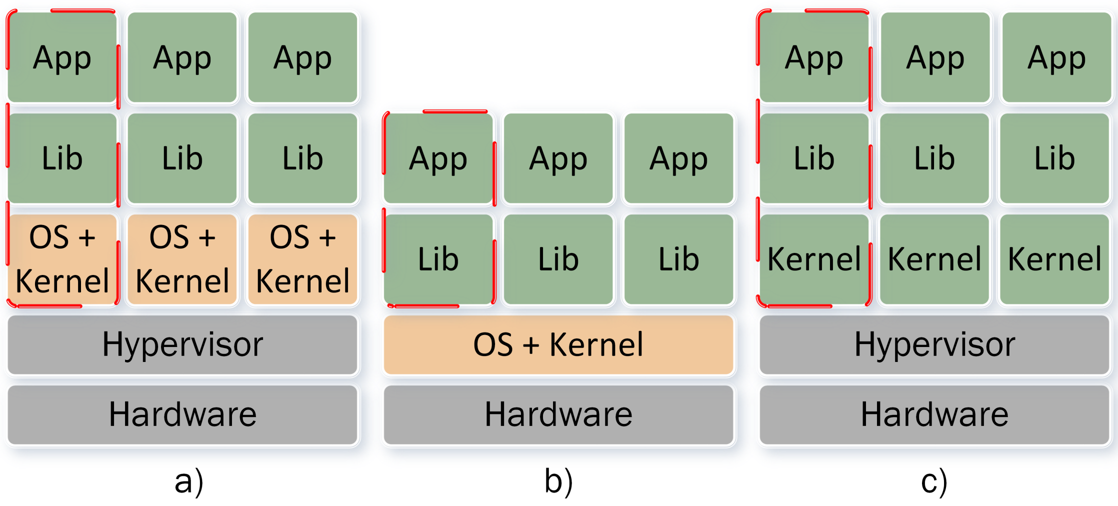

Figure 2.3: Comparison of a) virtual machine, b) container and c) unikernel system architecture

2.8 Virtualization technologies

In a traditional environment, the memory is divided in the kernel and the user space, for memory and hardware protection. The operating system (OS) and kernel are executed in the kernel space, while the libraries and the application software are running in the user space. A virtual environment follows similar principles. In some cases, though, a hypervisor is used for the virtualization of the underlying physical resources or the kernel is deployed to the user space. Figure 2.3 demonstrates the system architecture of each virtual environment, namely the VM, container and unikernel. The grey color illustrates the hardware or hypervisors, the orange indicates the kernel space and the green designates the user space.

2.8.1 Virtual machines

VMs are virtual environments that act as virtual computers with their own CPU, memory, storage and network interfaces, and have their own OS and kernels. A VM includes both user and kernel space, while the virtualization and allocation of the underlying physical resources is enabled by a hypervisor software (Fig. 2.3-a). VMs can support resource demanding applications and isolate them properly over the same infrastructure, but their instantiation time could be up to few minutes, depending, among other factors, on the footprint of the VM.

2.8.2 Containers

Containers can also provide an isolated environment for the applications and their runtime dependencies to run, using the common infrastructure. Contrary to the VMs, containers include only the user space where the applications and the library files reside, while the kernel space is shared (Fig. 2.3-b). Containers are more lightweight, compared to VMs, but they require an underlying OS and kernel that provide the basic services. Their instantiation time could be up to few seconds and they are mostly used to host lightweight applications.

2.8.3 Unikernels

Unikernels can be considered as shrunken VMs, as they contain the minimum required OS services and dependencies for a single task to run. They do not require a kernel space, as the kernel commands are executed in the user space, and they do not need any underlying OS; they can be executed directly at the hypervisor or on bare metal (Fig. 2.3-c). Since they contain only the essential information for the application to run, the size of the unikernels is very small, while the reduced kernel complexity makes the unikernels to run faster than the VMs. Thus, they can be instantiated almost instantly. In contrast, unikernels can only run a single process and the development of applications able to run on this environment is trickier. Finally, unikernels are prone to hack attacks, since their kernel uses the user space.

2.9 Key enabler virtualization and NFV orchestration tools

In this section we discuss the open-source or proprietary key enabler tools in terms of virtualization and NFVO. Openstack [37] and VMware [38] are two private cloud platforms that virtualize and manage the underlying physical resources in order to provide mainly VMs. Amazon web services (AWS) is a public cloud provider platform that provides VMs on-demand, over the Internet, while kubernetes [39] is an open-source system for automating deployment, scaling, and management of containerized applications. Openstack, VMware, AWS and kubernetes are considered as VIMs. Open source MANO (OSM) [40] and the open network automation platform (ONAP) [41] are two NFVOs, aligned with the ETSI NFV architecture. They interact with the VIMs and command them in order to execute the VNF and CNF LCM-related actions.

2.9.1 Openstack

Openstack is an open-source infrastructure-as-a-service (IaaS) platform which aims to provide VM services using the cloud computing principles, in generic purpose computers or servers. Countless adaptations, constant changes and updates by worldwide developers have lead to providing quality services for both educational and commercial purposes. Certain concepts and deployment models could be used based on specific needs, although alterations of these models are easy to occur. Openstack is managed by the Openstack Foundation, a non-profit organization which oversees both development and community-building around the project.

Openstack is divided in control and compute services, or openstack components. These components are responsible to provide the required tools for managing and executing the virtualization of the underlying physical resources, in terms of compute and network. There are two main deployment modes for openstack, the single-mode and the multi-node architectures. The single-node architecture describes the scenario at which all necessary openstack components are installed in a single physical computer or server, and has the benefits of low equipment, maintenance and operational costs, but with limited resources, flexibility and the resilience. The multi-node architectures require at least two physical machines, namely the controller and the compute node (or hypervisor), separating the openstack components, respectively.

Openstack natively supports horizontal scaling, just by adding extra compute nodes to the system, without affecting the system's integrity and availability. Additionally, it supports live migration, the concept at which VMs and are transferred among the hypervisors, unattended, in real time, and without service interruption. Finally, its open-source nature and its low hardware footprint, make openstack suitable for small scale environments and testbeds.

2.9.2 VMware

VMware is a proprietary-licensed cloud computing virtualization platform, designed to run on a special purpose hardware, such as blade servers and storage arrays. It follows a centralised architecture, where control services run on their own dedicated nodes while compute, network and storage resources are provided independently by other nodes. VMware is currently owned by Broadcom, focuses mainly on enterprises, while an academic license is also available.

Similar to openstack, VMware supports horizontal scaling, just by adding extra nodes. Additionally, a zero downtime live migration is supported, where workloads can be migrated from one server to another for maintenance, resilience and performance reasons. The VMware Telco Cloud Platform is a solution that supports both CNFs and VNFs and accelerates the deployment of 5G services by removing integration challenges between the platform and network functions, therefore making it a popular platform in the telecommunication industry.

2.9.3 Amazon web services

AWS is Amazon's public cloud computing platform that provides on-demand VMs and APIs in a pay-as-you-go manner. The distributed services are provided using AWS server farms and they include the Amazon elastic compute cloud (EC2), that provides vCPUs and vRAM, and the Amazon simple storage service (AS3) that provides storage services. Since it is a public cloud solution, the CAPEX investment is zero for the enterprises, and they pay only a pay-as-you-go OPEX cost. On the downside, due to the central locations of the AWS datacenters, a high latency can be observed, prohibiting the accommodation of latency critical 5G use cases.

To solve this, Amazon created a private MEC platform, namely AWS Outpost, which integrates edge computing infrastructure with private networks deployed on or near the customer's premises, eliminating the backhaul channel connection that adds latency between the UE and the application that runs on VMs. AWS Outposts addresses these challenges by providing a secure, dedicated cloud computing platform and reliable on-premises wireless networking based on 5G. MNOs have the ability to collaborate with AWS in order to provide private MEC solutions for enterprises. In this case, an initial CAPEX investment is needed, apart from the pay-as-you-go cost for accessing and managing the local AWS services. AWS' target audience are small, medium and big enterprises.

2.9.4 Kubernetes

Kubernetes (k8s) is an open-source system for automating deployment, scaling, and management of containerized applications. K8s consists of nodes (i.e., VMs, physical machines or cloud clusters) and multiple nodes comprise a k8s cluster. K8s executes the services by placing containers into k8s pods that run on k8s nodes. K8s clusters allow the orchestration and monitoring of containers.

K8s offers service discovery and load balancing functionalities, where the load is distributed among the k8s nodes. Additionally, the self-healing functionalities automatically replace the malfunctioning containers with operational ones. Automatic vertical scaling is also supported, where containers are allocated more or less resources, based on their current needs, while horizontal scaling can be achieved by adding more k8s nodes to the k8s cluster. On the contrary, container migration is tricky, due to the dependence of the containers from the underlying OS and kernel. To overcome this limitation, in the case where containers run in VMs, the whole VM may be migrated, migrating, thus, all the containers that run on it. This increases the orchestration complexity, in the case where multiple CNFs are running in a VM and not all of them need to be migrated.

Since the hardware footprint of k8s is relatively low, it can be used by small scale environments, as well as by more big and complex schemes. Being an open-source software, it also allows its use for academic and research purposes. Minikube [42] is a local k8s cluster that can be deployed on general purpose hardware, which can be used for learning and experimenting, while there is also a more lightweight version of kubernetes [43], namely the k3s, created by Rancher Labs for small-scale developing purposes.

2.9.5 Open Source MANO

OSM is developing an open source MANO stack aligned with the ETSI NFV architectural framework that was explained earlier, mainly deployed by Telefonica. OSM manages the i) VIM for the deployment and onboarding of the VNFs, as well as the virtual links that are connecting them, ii) Day-0, Day-1 and Day-2 configurations of the VNFs iii) autoscaling functionality of the VNFs iv) termination of the VNFs at the end of their lifecycle.

OSM supports descriptor files written in yaml, namely the VNF descriptor (VNFD) and the network service descriptor (NSD). The VNFD file is responsible for the instantiation parameters and operational behaviors of the VNFs, such as the virtual CPU (vCPU), the virtual RAM (vRAM) and the storage allocated per VNF (Day-0 configurations), the network initialization (Day-1 configuration) and the autoscaling threshold configuration (Day-2 configuration). The NSD file describes the internal or external connection points and their corresponding links that each VNF uses to communicate with other VNFs or with external networks. The orchestrator can construct VNF chains (VNFFGs or SFCs) by connecting the connection points.

OSM supports interfacing with both private (e.g., openstack or VMware based) and public (e.g., AWS based) cloud provider VIMs, for managing and orchestrating VNFs [44]. Additionally, OSM natively supports the interaction with cloud-native infrastructures (i.e., kubernetes), for managing and orchestrating CNFs. Being aligned with the ETSI NFV architecture, OSM increases the likelihood of interoperability among NFV implementations. The variety of LCM actions it supports, its open-source nature, as well as its low hardware requirements, make OSM a perfect candidate for testing in research environments.

2.9.6 Open network automation platform

ONAP is an open-source software platform that enables the design, creation, and orchestration of network and edge computing services and was developed mainly by The Linux Foundation. ONAP provides efficient, E2E infrastructure management and delivers automation on different on-demand services.

ONAP consists of two architectural frameworks, namely the design-time and the run-time. The design-time framework is responsible for the service design, that allows the modeling of the resources and relationships that make up the service, the policy rules that guide the behaviour of the service and the applications of the service. The planning of the VNF onboarding, the resource creation that compose the services and the services distribution are the main tasks of the service design. In regard to the run-time framework, the main tasks include the definition of the VNFs responsible for the service, the definition of the orchestration steps, the selection of the valid location where to be deployed, the VNF onboarding and instantiation, as well as the configuration of the VNFs

ONAP supports the interconnection with multiple VIMs, such as Openestack, VMware and AWS, but also supports the interfacing with kubernetes clusters. Its capabilities include the deployment, the configuration, the monitor, the healing, the scaling, the upgrade and the termination of the VNFs or CNFs it manages. The hardware footprint of ONAP is relatively big, making it difficult for use in small-scale testing environments.

Chapter 3

Fog-enabled Scalable C-V2X Architecture for Distributed 5G and Beyond Applications

3.1 Introduction

The IoT ecosystem is a collection of billions of devices, such as sensors, that are connected among them and with the Internet. According to Ericsson's mobility report, the 10.8 billion IoT connections of 2019 are expected to reach the number of 24.9 billion by the end of 2025, which means a compound annual growth rate (CAGR) of 15% [45]. When it comes to cellular networks, and as the 5G is being gradually introduced, the amount of 1.3 billion cellular IoT connections of 2019 is expected to experience an even higher CAGR of 25%, reaching the number of 5 billion by the end of 2025.

Vehicle-to-everything (V2X) communications is a major area of IoT that will enable communication between vehicles and between vehicles and infrastructure. The cellular V2X (C-V2X) application layer model includes the vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) and vehicle-to-network (V2N) operational modes that require a combination of high reliability and low latency. C-V2X was initially introduced by 3GPP, under the release 14 [46], using LTE-based RAN for V2X communications, while release 16 features 5G support for the V2X services [47].

The national highway traffic safety administration of the United States (U.S.) predicts that by fully adopting only two V2X safety applications, 1000 lives per year could be saved and half million crashes could be prevented in the U.S., while a reduction of 14% of the global greenhouse emissions, due to transportation, could be also achieved [48]. Therefore, it is imperative to create an intelligent transportation system (ITS), or an Internet of vehicles (IoV) environment, in order to provide crucial and non-crucial services.

Traffic safety, reduced air pollution and regulation of vehicle traffic flows, are only few examples for improving the quality of life. Traffic management applications, for example, could collect real-time weather information from road condition sensors, such as surface conditions (e.g., temperature, humidity, salinity, and so on.). A traffic safety application will have to take into consideration the aforementioned metrics and, along with sensors on the vehicles (e.g., proximity or break sensors), issue an action that could be a warning (e.g., information for cooperative road safety), or an immediate action (e.g., emergency break for collision avoidance).

Depending on the service, V2X crucial services (e.g., autonomous driving) can tolerate a maximum latency between 10 ms and few seconds, while non-crucial services (e.g., vehicle software update) can tolerate up to few minutes of latency [49]. LTE RAN is unable to support such low latency values in big volumes, which makes the 5G RAN the perfect enabler for C-V2X applications [50]. The majority of the published works propose edge computing solutions where processing power becomes available at the edge of the network. These proposals often refer to MEC-enabled architectures [51], where processing power is offered both at the core and edge network. Moreover, applications, in accordance with the NFV paradigm (i.e., the virtualization of the physical infrastructure), are considered as VNFs that can be executed at these two layers [52]. While these solutions are able to offer the required low latency for many C-V2X applications to function properly, they struggle to guarantee the ultra-low latency requirements (i.e., equal or less than 10 ms) in big volumes for some crucial services [50].

Addressing some of the core and edge limitations, fog computing "distributes computing, storage, control and networking functions closer to the users, along a cloud-to-thing continuum" [10]. Fog takes advantage of the infrastructure that lies along the cloud-to-thing path, such as servers, routers, switches, vehicles and smartphones. These devices, previously limited to operate autonomously, can now participate in a connected environment that efficiently manages their resources. Therefore, the ability to leverage all the available resources, especially at the thing (or UE) proximity, can guarantee the ultra-low latency in big volumes that the C-V2X applications require.

Within the context of the fog computing paradigm, we propose a new 3GPP compliant fog-enabled architecture with three different layers of processing power; the core, the edge and the vehicle. Our architecture, apart from the V2N communications, supports direct V2V communications, where, combined with the processing power at the vehicle, enables the execution of ultra-low latency applications, saving, at the same time, resources for the V2N communications. A distributed application model is adopted, where applications consist of virtual environments (i.e., VNFs) that can run as VMs, containers or unikernels. The combination of one or more virtual environments, distributed across the three different processing layers, creates an application-as-a-service function chain (AaaSFC).

Supporting this architecture, we explain two important LCM functionalities, the live migration and the horizontal scaling of the VNFs. Furthermore, we provide an experimental 5G platform implementation, based on open-source software and common hardware, while we enhance the shortcomings of an open-source NFVO in order to support migration decisions. Additionally, we present four distinctive examples of C-V2X use cases. To the best of our knowledge, there is no other work that assumes such distributed architecture, utilizing heterogeneous virtual environments, in order to deploy, place and manage distributed AaaSFsC within the premises of a three-layered fog-enabled environment for C-V2X use cases, supported by experimental results that are based on an experimental platform implementation.

The remainder of this chapter is organized as follows. Section 3.2 presents the related work. Section 3.3 provides four different C-V2X use cases. Section 3.4 discusses the fog-enabled C-V2X architecture and the lifecycle management. Section 3.5 delivers two C-V2X applications-as-a-service function chain examples. Section 3.6 describes the implemented 5G experimental platform. Section 3.7 analyzes the obtained experimental results, whereas section 3.8 is devoted to the chapter's conclusions.

3.2 Related work

A significant topic that has caught both industry's and academia's attention is the VNF placement problem in distributed architectures. On the one hand, there are various works that try to tackle the placement problem within a generic MEC-enabled environment considering a VNF either as a VM [52], an environment that fully virtualizes a physical computer, or a container [53], a lightweight virtual environment that enables application-level virtualization. Compared with unikernels, an even more shrunken virtualization environment that contains only the minimum amount of OS services, kernel and libraries for a specific application to run, VMs and containers could add significant delay to the IoV infrastructure, making them inappropriate for ultra-low latency C-V2X services.

On the other hand, there are very few works focusing on the service placement, specifically for C-V2X communications. The authors in [54] consider the C-V2X service placement problem in a MEC architecture, taking into consideration the compute resource availability at the nodes, offering a low-complexity greedy-based heuristic algorithm in order to solve this problem. Yet, they are limited to a two-layer architecture and their building block is considered to be the VM. Furthermore, the authors in [55] provide a fog-enabled platform in order to support the distribution of IoV applications. While they leverage the fog infrastructure, they are also limited to the container as their virtualization environment.

3.3 C-V2X use cases

The IoV aims to provide crucial and non-crucial applications for an ITS. While in the V2V, V2N and V2I operational modes the 5G base station (gNB) can be used as the communication hub, there is also the option for direct communication between the vehicle and another vehicle or device, using the PC5 interface. This direct channel can be used when the latency requirements of a service would be violated if the intermediate gNB was used, or to save infrastructure and network resources. In this section we will describe four C-V2X use cases, as well as their latency requirements, based on [49] and [56].

3.3.1 Vehicle type warnings

A vehicle type warning service could be on emergency situation to create awareness of the presence of emergency vehicles in the proximity. This application allows a vehicle of an emergency service (e.g., ambulance, police car, etc.) to indicate its presence (event). The emergency vehicle notifies the vehicles on its path about its presence (notification) and the drivers should be warned to clear the road (action). This will save important time for the emergency services to reach their destination. The maximum latency for such application is 100 ms, from the occurrence of the event until the notification of the involved vehicles [56].

3.3.2 Co-operative road safety

An emergency breaking application could be considered as a co-operative road safety service. The vehicle that uses a hard break (event) signals the hard breaking to the vehicles that are following it (notification) in order to notify them for possible collision. Based on the speed, distance and road conditions, the following vehicles should decide if they should slow down in order to avoid a possible collision (action). The maximum latency for such application is 100 ms, from the occurrence of the event until the decision for action of the involved vehicles [56].

3.3.3 Navigation and traffic jam avoidance

A map download and update application can be considered as an infotainment service. The vehicle requests Internet access in order to download the required map and find the best route, based on its current location and destination. The maximum latency for such application is 500 ms [56]. Traffic jams normally happen in a long time period. In case a traffic jam occurs (event) on the navigation's predefined route, depending on the setting, a delay of 2 seconds, in the case of urban areas, up to few minutes, in the case of rural areas, can occur for the vehicle to get notified (notification) and change its route (action) [49].

3.3.4 Autonomous driving

An autonomous driving application is enabled by HD sensor sharing. It provides mechanisms for vehicles to share HD sensor data, such as lidar that measures the

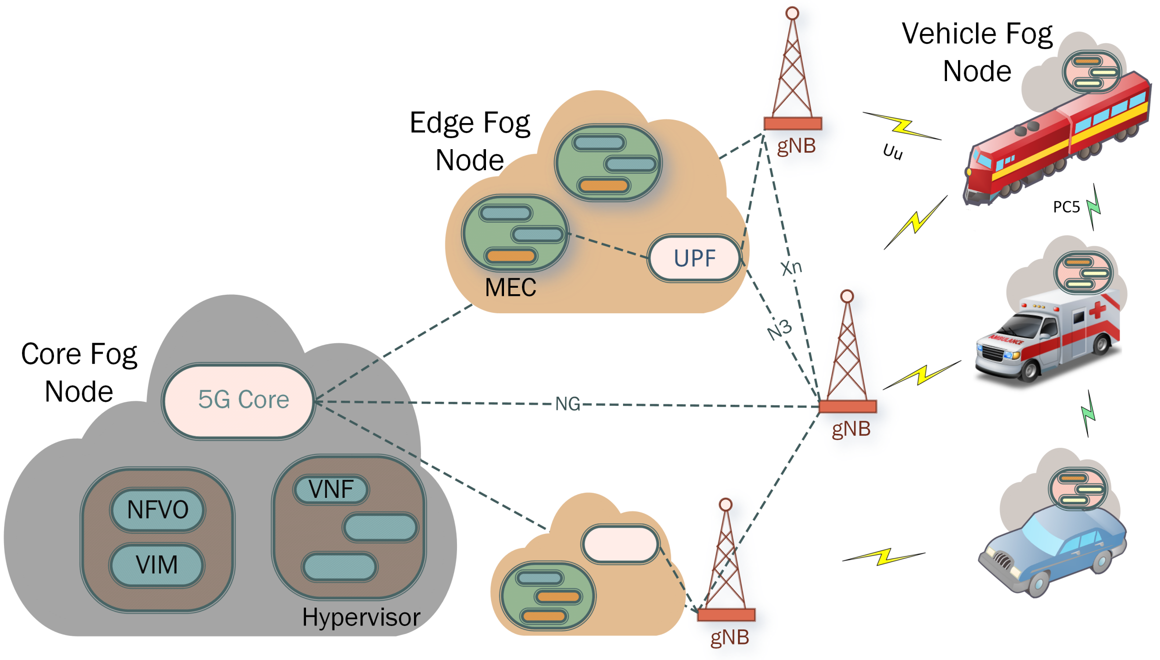

Figure 3.1: Fog-enabled C-V2X architecture

distances by illuminating targets with laser light and HD cameras, in order to enable better coordination for platooning and intersection management. An emergency vehicle that has to make a turn into an intersection (event) signals its course and speed to all nearby vehicles (notification) in order to adjust their speed to give priority to the emergency vehicle (action). The maximum latency that such application can suffer cannot exceed 10 ms, from the occurrence of the event until the actions taken by the involved vehicles [49].

3.4 Fog-enabled C-V2X Architecture

We propose the fog-enabled C-V2X architecture depicted in Fig. 3.1. This architecture includes computation, network and storage resources on each fog node, with the core having the most, and the vehicle the least resources. The MEC resides in the edge fog node. The proposed architecture fully supports NFV by enabling the virtualization of physical resources at the hypervisors that are located at the three nodes. Additionally, a distributed application model is adopted, where applications are hosted in connected VNFs. VNFs can run as VMs, containers or unikernels, while their combination creates an AaaSFC. The VIM is responsible for the management and control of the heterogeneous resources of the NFV infrastructure, while the NFVO performs the resource orchestration. They both reside in the core fog node.

In this architecture, the 5G RAN is considered for the communication of the three different fog nodes. The 5G core control functions reside in the core fog node and the UPF that steers the data traffic towards the desired applications and network functions resides in the edge fog node. The gNBs communicate among them using the NG interface, with the UPF using the N3 interface and with the vehicles using the Uu interface. The vehicle fog nodes can also communicate between them using the direct PC5 interface. The PC5 interface, as described in [47], enables the direct communication where the infrastructure does not participate. Finally, each vehicle fog node acts as an IoT gateway, as it collects and aggregates the data from the sensors deployed at the vehicle.

In terms of virtual environments deployment, the VMs can run either at the core or edge fog nodes, the containers at the edge or vehicle fog nodes and the unikernels are able to be executed at the vehicle fog nodes only. Regarding their connectivity, the VMs of the core fog node can communicate with VMs of the core and edge fog nodes, and with containers of the edge and vehicle fog nodes. Furthermore, the containers of the edge nodes can also communicate with the containers of the vehicle nodes. Finally, containers of the vehicle fog nodes can communicate with the unikernels and with containers on other vehicle fog nodes, while the unikernels can only communicate with the containers and sensors of the vehicle fog nodes. All the communications are enabled by virtual links.

There are numerous applications that can be executed in this architecture. Crucial services, such as autonomous driving could be combined with non-crucial services, such as traffic jam avoidance or infotainment services, running in parallel. In some cases, the initial allocation of the VNF resources can prove to be insufficient. Therefore, LCM actions, such as live migration and scaling, need to be performed to ensure a smooth service operation.

3.4.1 Lifecycle management

Each individual VNF has a lifecycle, which is governed by the NFVO that can be considered as the central controller of the system, in terms of filtering the incoming requests and (re)allocating the physical resources. The NFVO executes periodic checks in order to monitor the current availability of resources and ensures that the NFV infrastructure adapts to data traffic variations. Moreover, the NFVO is responsible for two important actions, namely the live migration and the scaling.

- **Live migration:** the process that involves moving a VNF to a different hypervisor for resource optimization purposes, without service interruption [27]. This is achieved by running the instances of the old and new hypervisor simultaneously while service migration is performed, and only migrating RAM contents as a final step.

- **Scaling:** the ability of the VNFs to scale-out upon increased resource utilization and scale-in upon resource underutilization. In this architecture, the scaling option that is selected is the horizontal scaling (i.e., the instantiation of more VNFs). Compared with the vertical scaling (i.e., the increase of the resources allocated to a VNF), the horizontal scaling provides the distributed applications with the required elasticity, redundancy and continuous availability.

3.5 C-V2X applications-as-a-service function chain

In this section, we provide two examples of how C-V2X use cases can be developed as AaaSFCs, as well as their LCM.

3.5.1 Co-operative road safety

Let us consider the co-operative road safety example, combined with a vehicle warning feature that were described earlier. If we want to deploy it as a simplified AaaSFC (Fig. 3.2–dotted line), it involves:

- one VM module that runs at the core and hosts the traffic manager module (TM),

- one container module that runs at the vehicle, monitors the data of the unikernels and notifies possible involved parties (MON), and

- one unikernel module that runs at the vehicle where the data of the breaking and proximity sensors can be collected (BRP).

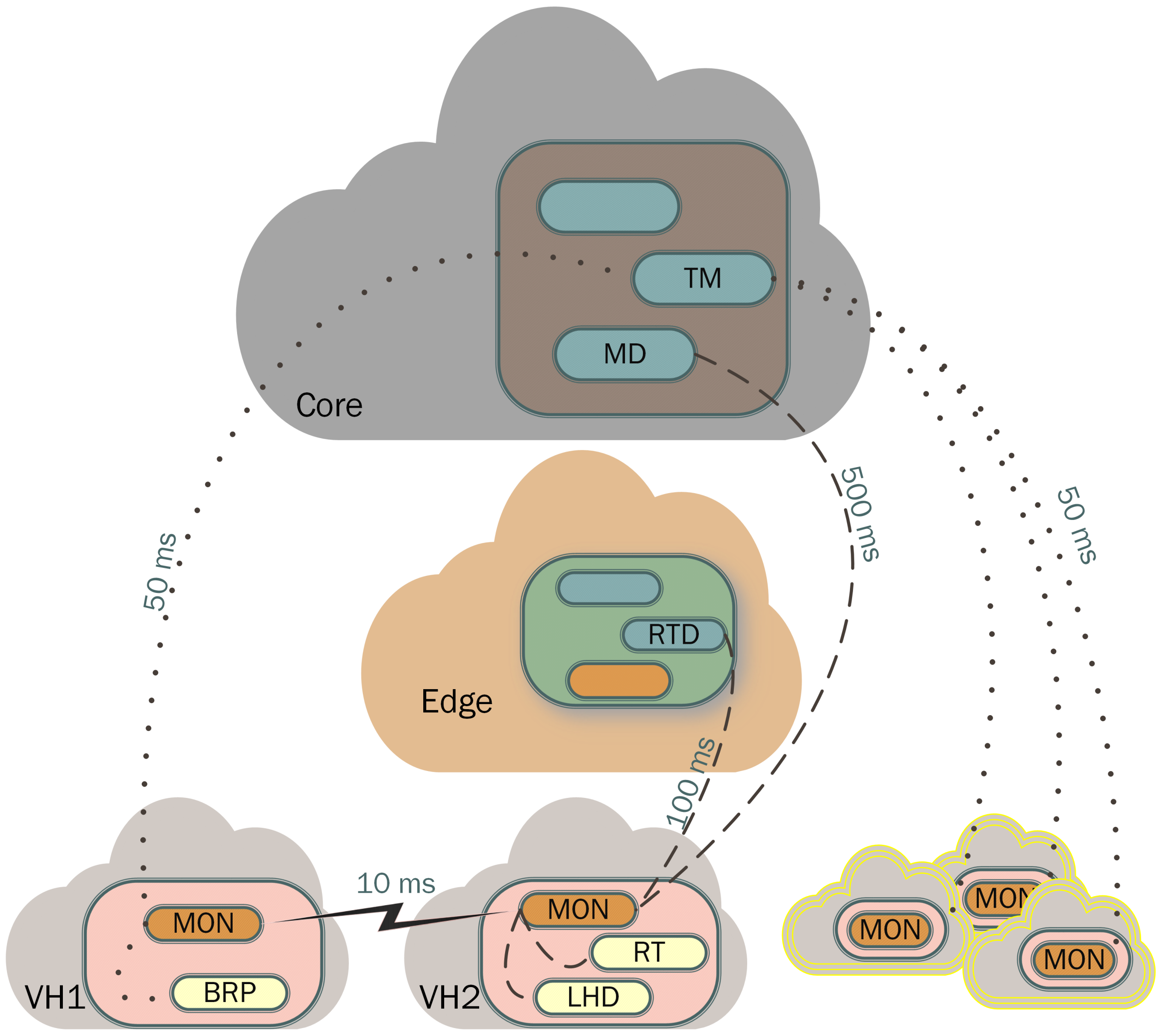

Figure 3.2: AaaSFC for co-operative drive safety (dotted line) and for autonomous driving (dashed line)

In case a hard break occurs (BRP) on vehicle 1 (VH1), the MON checks for any possible vehicles in proximity (VH2) and notifies them for the hard break (through the direct PC5 interface) in order for the latter to decide an emergency break action. At the same time, the MON sends the notification to the TM, through the 5G network, in order for the latter to issue and broadcast a warning notification to the vehicles that follow VH1, but are out of range for direct V2V communication.

3.5.2 Autonomous driving

Let us consider the autonomous driving application as part of many applications that cooperate. In a simplified version, we could involve two of the previously described applications, the HD sensor data and the navigation and traffic jam avoidance. If we want to deploy it as an AaaSFC (Fig. 3.2–dashed line), it involves:

- one VM module that runs at the core and hosts the latest map data of a geographical area (MD),
- one VM module that runs at the edge and monitors the real-time vehicle traffic data of an urban area (RTD),
- one container module that runs at the vehicle, monitors the data of the unikernels and notifies possible involved parties (MON), and
- two unikernel modules that run at the vehicle where the data of the lidar and the HD cameras are collected (LHD) and the route is determined (RT).

In case there is a request for an emergency service, the MON requests from the RTD the best route, based on real-time vehicle traffic data, and updates the RT with the route. When the vehicle arrives to an intersection, the data from the LHD will be used by the MON to notify all nearby vehicles using the direct PC5 interface. While on route, a traffic jam is detected. The RTD forwards an alternative route to the MON but the RT map data are not updated; thus, the MON requests the additional map from the MD and forwards it to the RT. In the last part, the edge acts as the intermediate in the vehicle-to-core communication.

3.5.3 AaaSFC lifecycle management

Each VNF has specific virtual resources allocated to it and each virtual link that interconnects the VNFs has a maximum latency that can tolerate, based on the application. Depending on real-time data traffic conditions, the application's SLAs might be violated. In what follows, we explain the conditions under which a migration or a scaling decision might occur, in order to prevent such violation.

3.5.3.1 Core to edge migration decision

The example of the co-operative road safety uses the TM/VM of the core. In case of high data traffic (e.g., during rush hours), the required for the service latency could be violated. Thus, we propose a migration action from the core to the edge, as shown in Fig. 3.3-a, in order for the TM/VM to be closer to the vehicles. In the

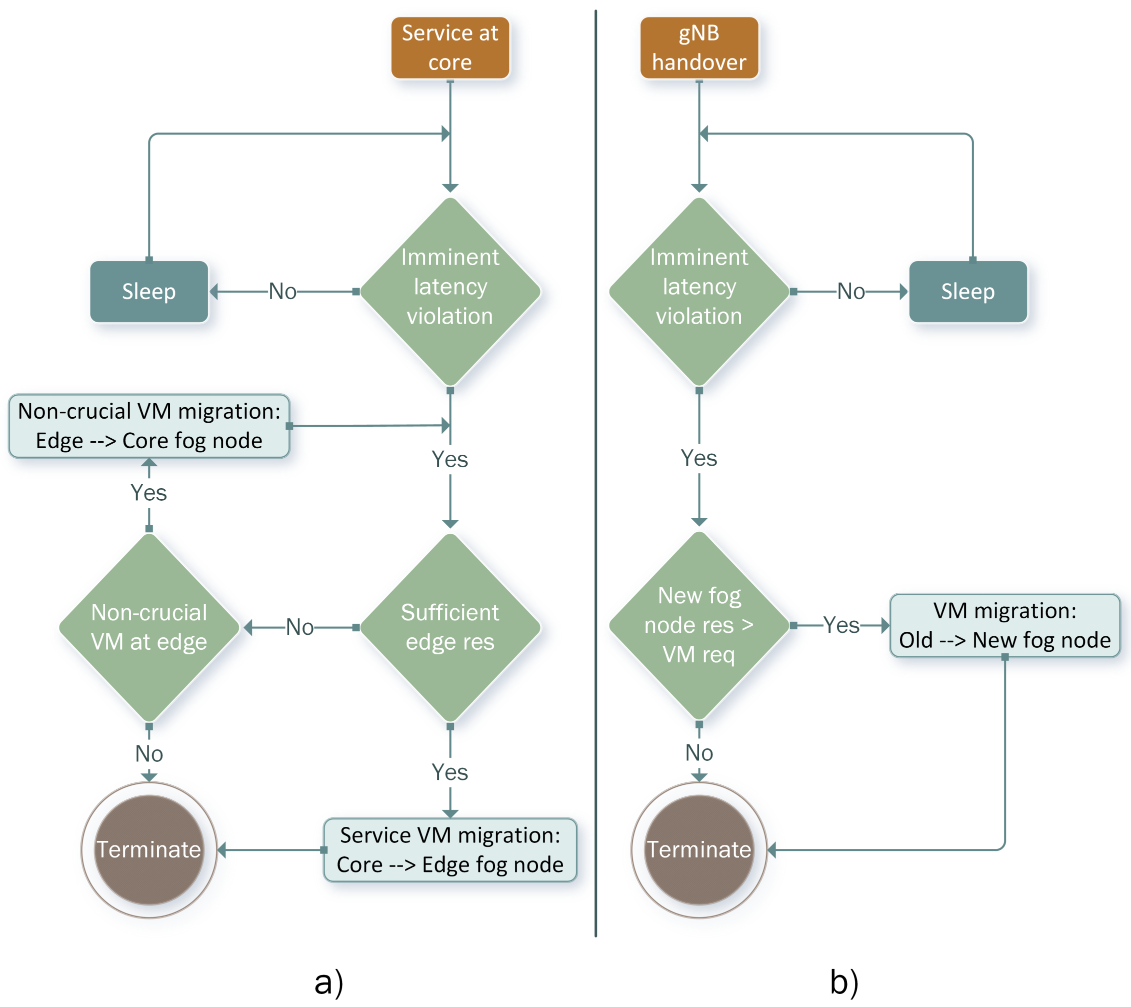
Service at core
Imminent latency violation
No
Sleep
Yes
Sufficient edge res
No
Non-crucial VM at edge
Yes
Non-crucial VM migration: Edge --> Core fog node
No
Terminate
Yes
Service VM migration: Core --> Edge fog node
a)
gNB handover
Imminent latency violation
No
Sleep
Yes
New fog node res > VM req
Yes
VM migration: Old --> New fog node
No
Terminate
b)

Figure 3.3: Core to edge (a) and edge to edge (b) migration decision flow charts

case of insufficient edge resources, a migration from the edge to the core of non-crucial VNFs could take place in order to free up the needed edge resources for the TM/VM accommodation. When the data traffic is restored to normal, the VNFs could be migrated back to their original hosts.

3.5.3.2 Edge to edge migration decision

Another action that could result in migration is due to the vehicles' mobility. Since the vehicles move from one gNB to another, if the two gNBs are not served by the same edge fog node, then a migration of the service is needed from the old to the new edge fog node, as described in Fig. 3.3-b, following the user's handover process between the gNBs. Since the edge fog nodes communicate between them, the

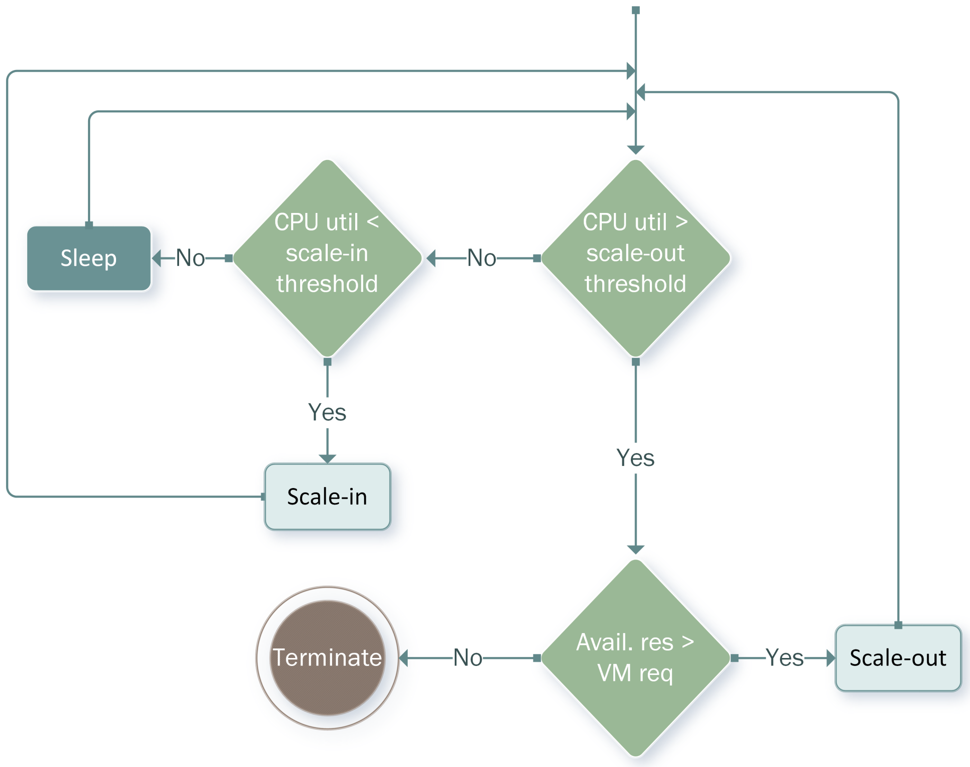

Figure 3.4: Scaling decision flow chart

VM could have remained at the old node. However, this would result in increased latency, since more hops are added between the vehicle-to-edge communication.

3.5.3.3 Scaling decision

In case of high data traffic, one or more of the VNFs that are part of the AaaSFC can reach their maximum CPU utilization capacity. In this case, as demonstrated in Fig. 3.4, the NFVO will decide a scale-out operation by creating a new copy of the VNF that exceeds the CPU utilization threshold (i.e., scale-out threshold), balancing the load between the two VNF instances. This process can be repeated until the data traffic can be handled properly, as long as there are sufficient physical resources. In the case of scaling, the total VNFs of an AaaSFC are consequently increased. The NFVO is also responsible for the reverse process, the scale-in, when the traffic is restored to normal and the CPU utilization threshold is met (i.e., scale-in threshold).

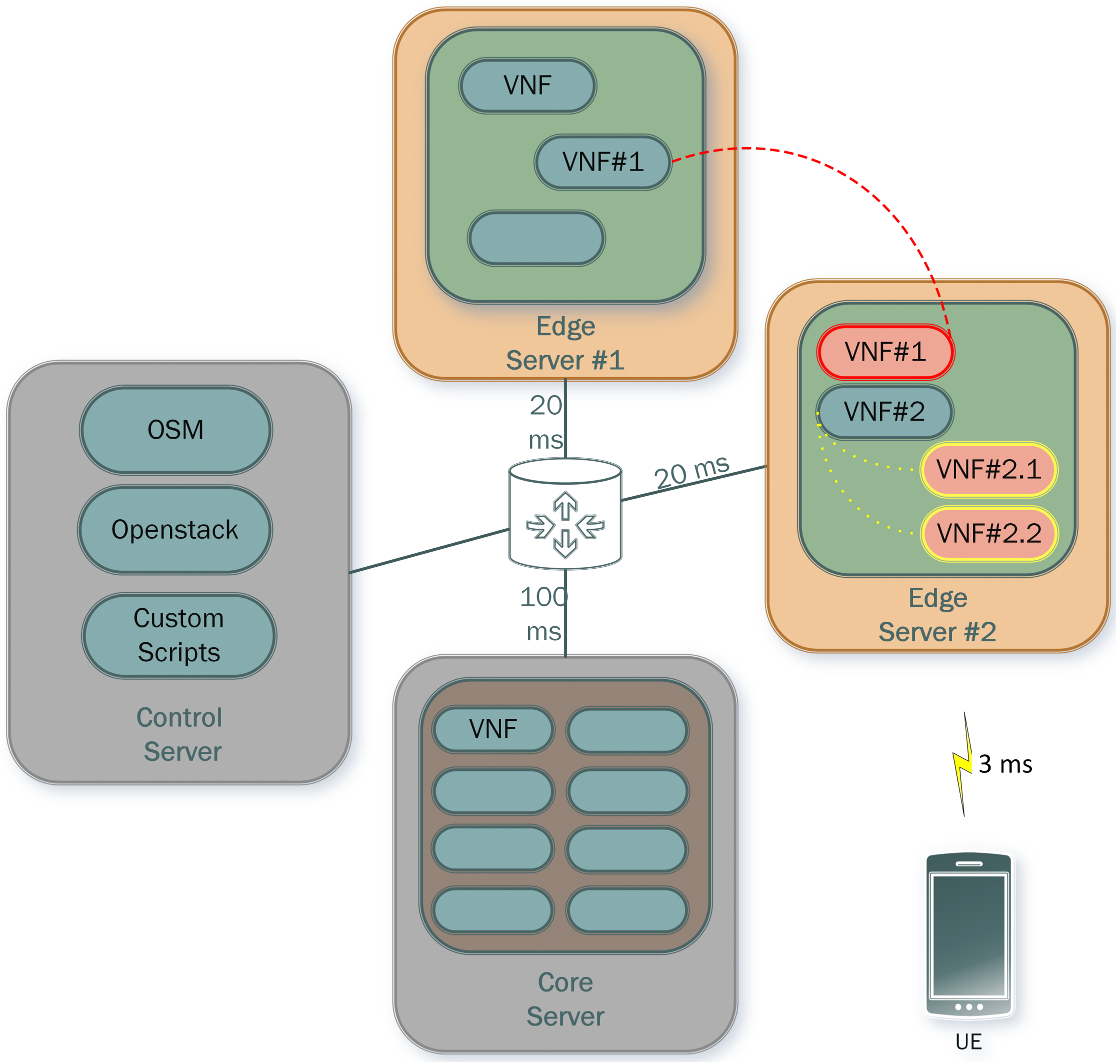

Figure 3.5: 5G experimental platform implementation

3.6 5G experimental platform implementation

In order to demonstrate a proof of concept of our architecture, we developed a 5G experimental platform (Fig. 3.5). The hardware of the platform (Table 3.1) consists of one control server for the management and orchestration of the physical and virtual infrastructure, and one core and two edge servers for the core and edge processing needs, respectively. In terms of compute resources, the physical servers at the edge site have lower compute power compared to the server at the core. In terms of networking, all physical servers have two network interface cards (NICs) and are connected to two routers through 1 Gbps ETH interfaces, while

Table 3.1: 5G experimental platform hardware specification

	Control Server	Core Server	Edge Server #1	Edge Server #2
CPU	Intel i5-8500	i5-8500	i5-7400	i5-7400
Cores	6	6	4	4
RAM	32GB	32GB	16GB	16GB
Storage	250 GB	2x 250 GB	120 GB	120 GB
NIC	2x1Gbps ETH	2x1Gbps ETH	2x1Gbps ETH	2x1Gbps ETH, 1x 802.11n

edge server#2 has an additional 802.11n interface where a smartphone that acts as the UE can be connected.

With respect to the software installation, Openstack [37], on its Queens release, is the open-source IaaS platform that is employed as the VIM, in order to deploy and control the VNFs, while OSM [40], on its sixth release, is the software that is used to enable the VNF management and orchestration. Finally, neither Openstack nor OSM are aware of the application that runs on the VNF or its SLAs. Therefore, we have deployed bash-based custom scripts that monitor the UE-VNF latency and trigger the migration decisions, based on custom-defined thresholds.

3.7 Experimental analysis

In order to validate the described architecture, we conducted a set of experiments, leveraging the experimental platform.

3.7.1 Experimental setup

In our setup (Fig. 3.5), we assume service#1 and service#2 that are hosted on the VNF#1 at the edge server#1 and on the VNF#2 at the edge server#2 respectively. As described in Table 3.2, service#1 has an SLA that can tolerate up to 100 ms of latency, while VNF#1 needs 5% of CPU resources in order to process each incoming request. Service#2 has an SLA that can tolerate up to 15 ms of latency and VNF#2 needs 9.8% of CPU resources in order to process each incoming request. Furthermore, if the CPU utilization of VNF#2 exceeds 90% of the CPU resources, the process time for each request becomes unstable and the latency SLA for service#2 is violated. Thus, the scale-out threshold is set at 88% of CPU utilization, while the scale-in at 33% of CPU utilization. In order to equally distribute the data

Table 3.2: Service - VNF characteristics

	Latency	CPU util per request	Scale-out threshold	Scale-in threshold
Service#1 (VNF#1)	up to 100 ms	5%	88%	33%
Service#2 (VNF#2)	up to 15 ms	9.8%	88%	33%

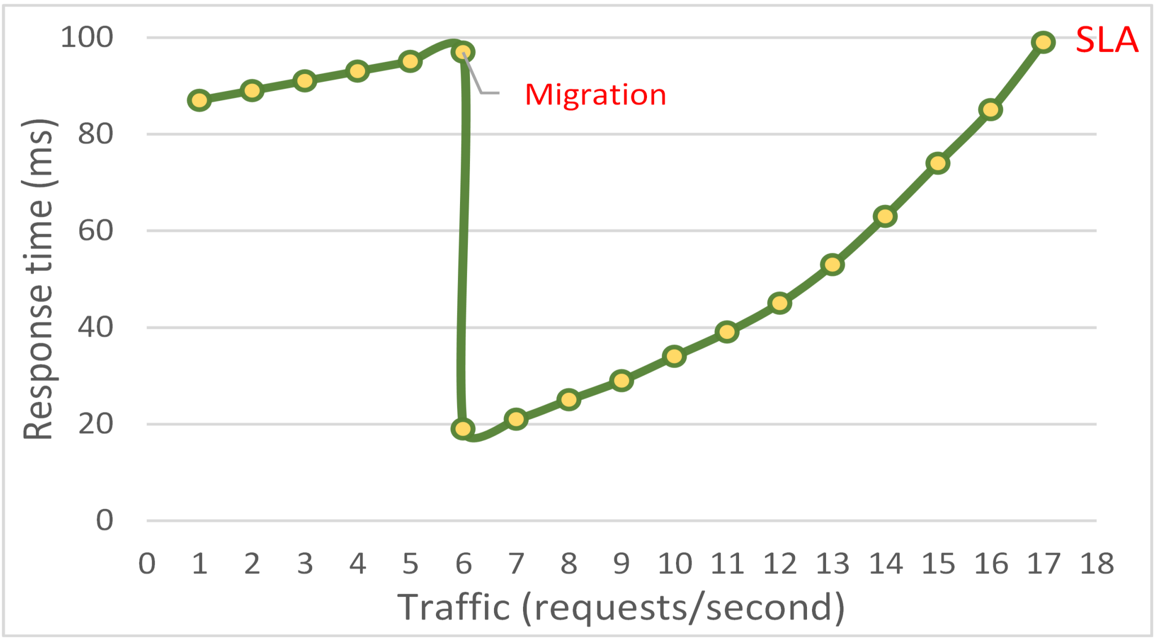

Figure 3.6: Demonstration of edge to edge migration functionality

traffic among the VNFs, we have already deployed a load balancer with round robin policy. Finally, we can define the number of requests/s that the UE, which serves as the vehicle, can send. For the live migration experiment, we start with 1 request/s and gradually increase to 17 requests/s. For the scaling experiment we start with 1 request/s and gradually increase to 27 requests/s. The duration that each experiment run was 12h.

3.7.2 Live migration experiment

The first experiment demonstrates the case of a vehicle leaving a gNB's range and entering another's, and the two gNBs are not served by the same edge fog node. Figure 3.6 depicts the response time versus the requests/s that the vehicle produces. Up to the first 6 requests/s, the VNF that serves this vehicle runs at the old edge fog node (i.e., edge server#1) and the service response time is 99 ms. If the vehicle makes more requests, the service#1 latency SLA will be violated. Thus,

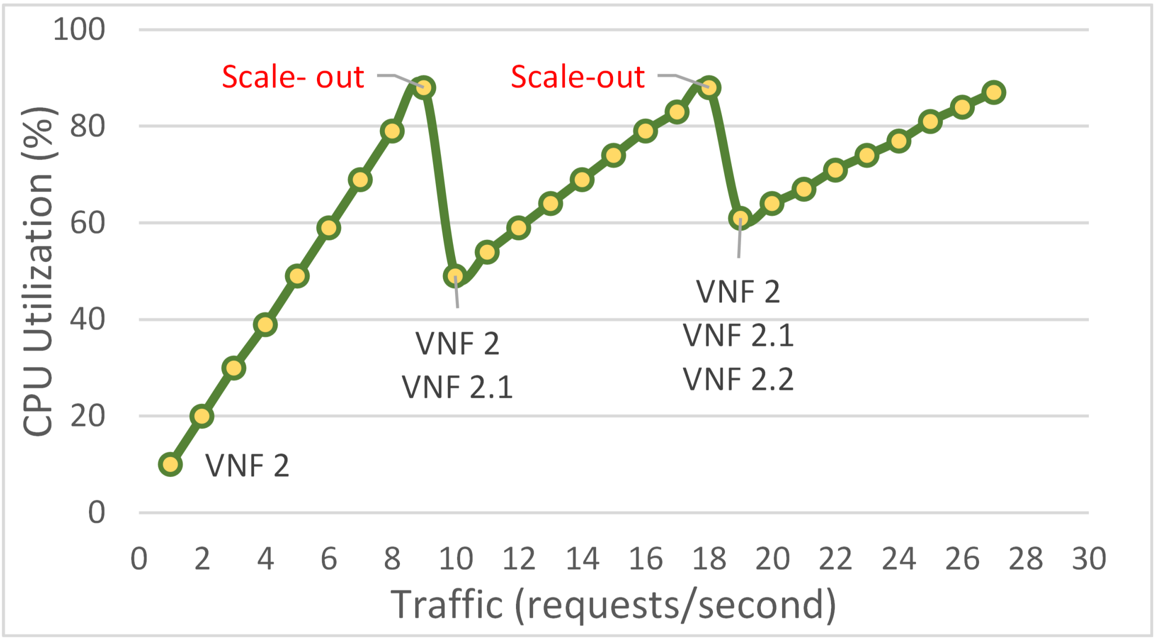

Figure 3.7: Demonstration of scaling functionality

the solution is to migrate the VNF#1 to the nearest to the vehicle edge fog node (i.e., edge server#2). This reduces the overall response time, since the delay that was added by the edge-to-edge server communication is now eliminated. A 183% increase can be observed in the accepted incoming requests, as from the originally maximum 6 requests/s, VNF#1 can support up to 17 requests/s when the migration action is complete. This way, we can support more requests without violating the latency SLA, compared with environments that do not support edge-to-edge migration functionalities.

3.7.3 Scaling experiment

The second experiment shows the case of increased demand for a specific service (i.e., during rush hours) but the VNF's already allocated resources cannot cope with such high demand. Figure 3.7 displays the average CPU utilization of the active VNFs versus the requests/s that the UE sends. Up to the first 9 requests/s, the process time at VNF#2 is stable, with a CPU utilization of 88.2%. Since the scale-out threshold has been exceeded, VNF#2.1 is instantiated. Hence, the load balancer equally distributes the data traffic and each VNF reaches approximately 45% CPU utilization. As the requests/s increase, the scale-out process is executed once more when the average CPU utilization of the two VNFs exceeds again the

88% threshold. This results in the instantiation of VNF#2.2. Once the data traffic is reduced, the scale-in process will destroy the VNFs that are underutilized. In this way, the resources can be efficiently managed and the application can serve more users when needed, compared with monolithic architectures that do not support scaling functionalities.

3.8 Conclusion

In this chapter, we proposed a fog-enabled C-V2X architecture, able to exploit the interplay among the core, the edge and the vehicle layers. We presented specific examples of C-V2X use cases that can be deployed as AaaSFC and provided information on their LCM. Additionally, we deployed an experimental platform, where open-source software, along with custom-made scripts, were used to demonstrate experiments that take advantage of the proposed architecture and validate the usefulness of the live migration and scaling features. As a result, a 183% increase of the accepted incoming requests upon mobility scenarios is observed, respecting the latency SLAs, while with the scaling experiments the served users were maximized.

Chapter 4

Online VNF Lifecycle Management in a MEC-enabled 5G IoT Architecture

4.1 Introduction

The exponential increase in requests for a variety of services creates the need for an omnipresent network, which should be faster, more responsive and reliable, and easily accessed under any conditions. According to recent reports, by 2024 the mobile subscriptions will reach 8.3bn, a number that exceeds the current worldwide population by 0.2bn [45]. Approximately, 45% of this cellular traffic is expected to be generated by an expanding ecosystem of smart connected devices, known as the IoT paradigm. The IoT growth is further accelerated by the penetration of IoT applications and services in our everyday life, as well as in a large segment of vertical industries, such as connected cars, smart homes, smart metering and industry automation [57].

The wide range of IoT services calls for a disruptive, highly efficient, scalable and flexible communication network, able to cope with the increasing demands and number of connected devices, as well as the diverse and stringent application requirements. For instance, UHD video streaming or augmented reality applications have increased bandwidth requirements, whereas autonomous driving, tactile Internet and factory automation require low E2E latency. In this context, the 5G of wireless communications, bringing together a set of enabling technologies, supports and advances the potential of the IoT technology.

SDN is one of the key enabling technologies that paved the road towards the 5G revolution, by permitting the replacement of specific network equipment, used in a dedicated way, with software that can be executed in generic purpose hardware, enabling the separation of the data and the control plane. Furthermore, the NFV technology [58] enables the virtualization of this networking software; hence, application and network functionalities are handled as VNFs and managed by an NFVO, able to have control upon the various locations of a distributed system [5]. The flexibility offered by the SDN/NFV network design is taken one step further with the network slicing paradigm, which enables the creation of multiple logical networks over a common physical infrastructure, offering the necessary isolation to support multiple 5G services with different requirements [6].

The virtualization of functions and the flexibility in their placement is highly aligned with the concept of MEC, proposed by ETSI [7]. MEC technology is defined as the cloud computing capabilities offered at the edge of the network, at the end users' proximity. MEC is responsible for delivering computing, storage and networking resources to the UE, thus achieving significant reductions in service response time and increasing reliability and security, since services are located much closer to the users, instead of a remote cloud. IoT is widely considered one of the key use cases of the MEC technology [7, 32]. First, a wide range of IoT services can be deployed to the edge, including IoT data aggregation services, big data analytics, video streaming transcoders, and so forth, ensuring low-latency and ultra-reliable performance. Second, IoT devices, which often have limited computational and storage resources (e.g., sensors, smart meters, etc.), can significantly enhance their capabilities by offloading tasks and services to the edge [59–66].

Even though MEC is clearly one of the major players towards the 5G realization and the future IoT services, the technology is still at its infancy [67], and challenges [68], such as efficient deployment, resource allocation and optimization, and application LCM, arise. Another non-trivial issue refers to the scheduling and placement of VNFs over the underlying infrastructure, including different MEC and remote cloud locations. In [69], cloud service deployment is modeled as a graph embedding problem, where service VNFFGs, or VNF chains, are embedded on top of a network of hypervisors (i.e., compute nodes). This is an NP-complete problem, and, hence, it is very time consuming to find an optimal solution even for small networks. However, reaction times in modern cloud-native infrastructures have gradually shortened to the point where services are individually scaled-out and scaled-in, responding to user demand in a matter of seconds. In order to keep up with the challenging NFV environments, service orchestration and scale operations must be orchestrated in real-time, with minimal computational complexity.

In order to efficiently manage the NFV ecosystem, there is the need for online and agile techniques for scheduling and orchestration of VNFs, as well as real environments that this technology can be applied to, in order to provide transparent and diligent testing and assessment. In this Chapter, we present a MEC-enabled 5G architecture, distributing the resources to the edge and core tiers. Furthermore, we propose two novel algorithms for the joint orchestration of the MEC and Cloud resources, thus enhancing the NFVO capabilities, while we test them at our custom-built experimental platform.

Specifically, we first present an algorithm for the VNFFG embedding of virtualized chained services, taking into account their latency requirements and service priorities (e.g., based on their criticality). Then, we propose another algorithm for the real-time allocation of the VNFs to the MEC and cloud resources, leveraging real-time service scale-out and scale-in features to meet the user service requests. Additionally, the second algorithm supports live service migration to further enhance the initial service placement, in order to efficiently handle the latency critical applications. Finally, we proceed to the validation of our proposed algorithm in a MEC-enabled 5G testbed implementation, deployed using open-source software over generic purpose hardware. The obtained experimental results, based on real-world 5G scenarios and cloud applications, provide useful insights for the potential of MEC-enabled architectures for real-life applications.

The remainder of this chapter is organized as follows. Section 4.2 describes the related works. Section 4.3 presents the overall NFV-enabled architecture, along with some key concepts and the considered system model. Section 4.4 provides the proposed orchestration algorithms for VNF onboarding and scheduling. Section 4.5 discusses the 5G testbed implementation and the employed open-source tools for its realization. Section 4.6 delivers the obtained experimental results, thoroughly explaining the different experimental scenarios, whereas section 4.7 is devoted to the chapter's conclusions.

4.2 Related work

Having a closer look in the SoA works regarding the VNF placement problem, its study can be performed through studying the placement and management of VMs [70]. Furthermore, we can find two different approaches in the literature: i) offline, where the placement decision is taken beforehand in order to satisfy end-users' requests, under various constraints, and ii) online, where the placement is happening live, taking into consideration real data, such as hypervisor load, to place the VMs [27]. On the one hand, concerning the offline works, authors in [15] present an advanced predictive placement algorithm where the optimal placement location is defined by the least used location that is closest to the majority of the UEs. In [16], a mathematical optimization model for VNF placement and provisioning is proposed, guaranteeing the QoS by including latency into the VNF chaining constraints. They focus, however, only on the placement of the virtualized LTE core functions, omitting the management and orchestration of the cloud applications and services that could be co-hosted in the same infrastructure. Authors in [17] study the dynamic deployment of chained network services on different VMs and formulate their reallocation of VNFs as a mixed integer program, focusing on server power consumption. The migration solution provided in this work, though, is applicable only in networks with repetitive, over a specific time interval, traffic scheme. One basic limitation of the aforementioned works is that the performance of the proposed algorithms is assessed only through simulations.

On the other hand, regarding the online solutions, in [18] the authors study how to deploy and scale VNF chains on-the-fly, using VNF replication across geo-distributed datacenters for operational cost minimization. Nevertheless, they limit each VNF chain deployment and scaling within the same datacenter. Furthermore, a traffic forecasting method for placing or scaling the VNF instances to minimize the inter-rack traffic is presented in [19], in the premises of a cloud datacenter. Even though a real implementation is offered, along with operator traffic driven simulations, the placement method in this work does not take into consideration the different SLA requirements each VNF might have. Finally, both works are limited within the premises of a datacenter, failing to exploit the potential benefits offered by edge-cloud architectures.

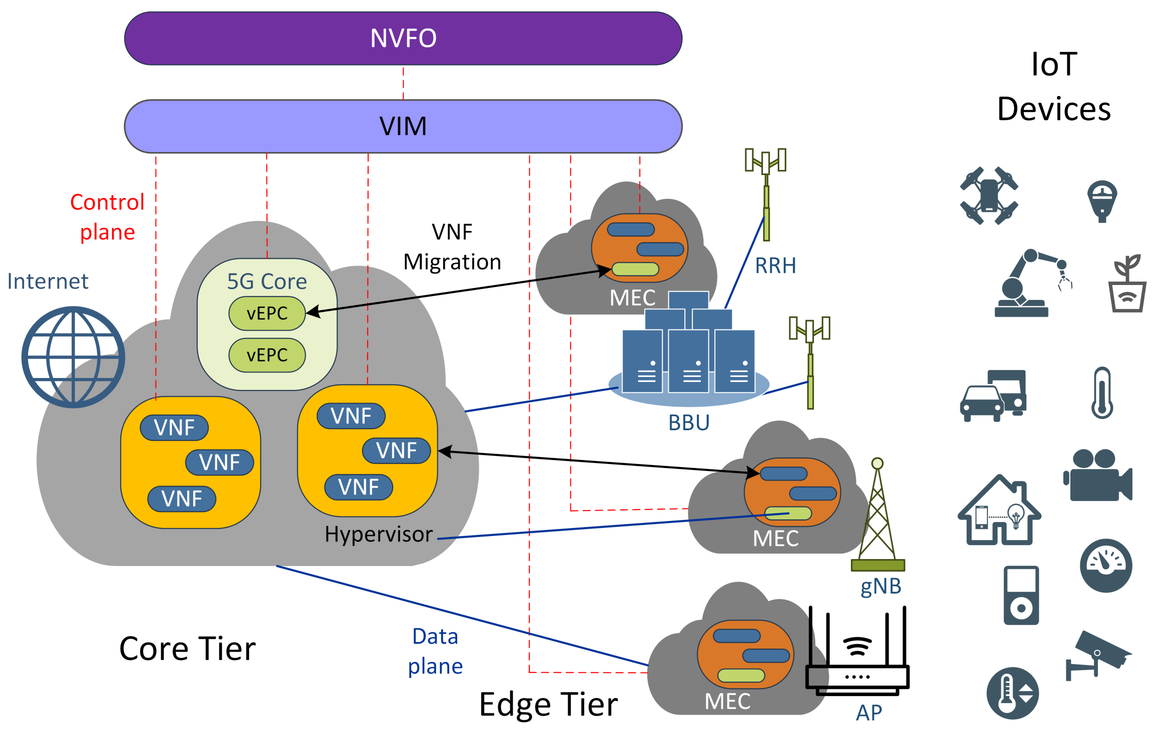

Figure 4.1: MEC-enabled 5G IoT architecture

On a different note, there is a very limited amount of experimental works in the literature that tackles with MEC implementations, where new challenges arise in order to deploy and orchestrate a programmable and flexible MEC-enabled 5G testbed. For instance, a 5G-aware proof of concept of an evaluation testbed with MEC capabilities has been described in detail in [71], without conducting though any real experiments to provide results. Furthermore, [53, 72] are based on containers, another virtualization technology, which share the host OS and provide process level isolation only, whereas their orchestrator has limited capabilities, without the ability to support migration features. Finally, these works are limited to the technical implementation of the testbeds and do not tackle the VNF placement problem. To the best of our knowledge, there is no related work that simultaneously: i) combines the interplay of the MEC with the cloud, in a virtualized manner, ii) proposes and implements an online VNF placement algorithm, iii) exploits VNF migration and scaling capabilities to meet the service demands in real-time, and iv) provides experimental results over a real-like 5G testbed experimental platform.

4.3 NFV architecture

We consider a MEC-enabled 5G IoT architecture depicted in Fig. 4.1. An heterogeneous RAN topology is considered for the connection of the IoT devices that may employ different wireless technologies. In particular, we consider a network that includes standalone gNBs, IoT access points (APs) and a C-RAN deployment, where BBUs are connected with RRH units. This architecture fully supports NFV by enabling the virtualization of compute and network resources at the MEC and cloud hypervisors, located at the edge and core tier respectively. The VIM is responsible for the management and control of the compute, storage and network resources of the NFVI, while the NFVO performs the compute and network resource orchestration.

4.3.1 Edge computing

The considered architecture includes two tiers of computational resources: the cloud at the core tier and the MEC at the edge tier. These are in the form of hypervisors (or compute nodes) where application and network VNFs are hosted for the duration of their lifecycle. Hypervisors are interconnected with an SDN data-plane, forming a leaf-spine topology, i.e., a mesh with a constant number of hops. Although different topologies have been considered in the literature, the leaf-spine is standardized in modern data centers as it simplifies VNF scheduling and guarantees a fixed latency for the data plane. It must be noted that the edge tier, which are the MEC hosts, contains limited computing resources. These are typically allocated to VNFs that should be placed closer to the UE to satisfy specific service requirements (typically low latency).

4.3.2 Virtual network functions

Fully adopting the NFV paradigm, we consider that the 5G cloud applications are implemented in the form of VNFFGs that result in VNF chaining. Each virtual link has its own bandwidth and latency requirements, which are typically encoded in the VNFD file, along with other VNF metadata. During the VNF placement, network slicing can be employed to guarantee the networking requirements of the VNFs. Network slicing ensures service isolation and offers performance guarantees to the service tenants by reserving appropriate resources as denoted by the

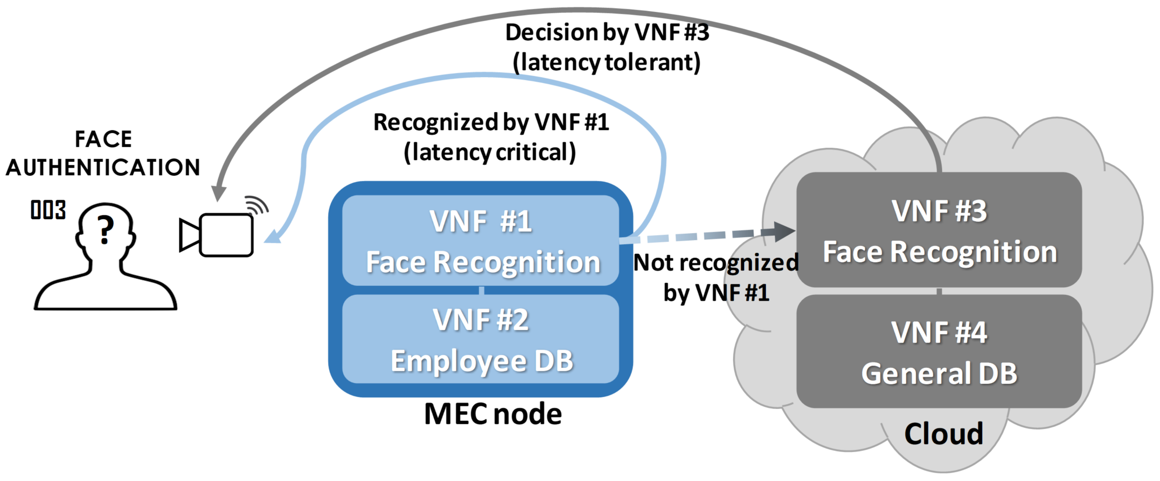

Figure 4.2: VNF chaining for face recognition

VNFD. Network slicing in 5G networks is supported by the programmable infrastructure, via appropriate northbound APIs. However, dedicated slices bear a significant cost for service providers, as resources are reserved even when they are not used by clients, hence negating any potential statistical multiplexing gains. Therefore, dedicated slices are typically associated with services with high QoS requirements.

Furthermore, based on their delay constraints, VNFs can be classified as latency-critical VNFs (LCVNFs), which are sensitive to latency, and latency-tolerant VNFs (LTVNFs) that can tolerate a higher degree of delay. Accordingly, the 5G cloud applications can be classified intro three categories: i) real-time applications, consisting of high priority LCVNFs (HP LCVNFs), ii) near real-time applications, consisting of low priority LCVNFs (LP LCVNFs), and iii) non real-time applications that consist of LTVNFs. The VNF chaining feature, though, allows us to combine and connect the aforementioned VNFs. In general, due to its limited resources compared with the cloud, the MEC entity is usually reserved for LCVNFs, which are placed in proximity to the UEs, in order to minimize latency. On the other hand, LTVNFs, or even LP LCVNFs in specific situations, can be safely deployed to the cloud.

An example of VNF chaining is given in Fig. 4.2. A set of VNFs is chained both in the same and in separated hypervisors, in order to identify a person at the entrance of a company. Although the face recognition application [73] is broadly known through the cloudlets edge computing concept, it is a use case that is also compatible with MEC [74]. To achieve faster response time for the employees of the company, a MEC node is deployed to the edge that hosts two chained services: i) a face recognition VNF, and ii) a database (DB) VNF. If the person is identified in the employee DB, the whole process is finalized. Otherwise, VNF#1 sends its output to the face recognition VNF in the cloud that is VNF#3, where the same procedure occurs with a general DB (VNF#4) including employees of the company from other locations, or customers.

4.3.3 VNF lifecycle

Each individual VNF has a lifecycle, which is controlled and managed by the NFVO. The NFVO resides in the core tier and can be considered as the central controller of the system, in terms of filtering the incoming requests and (re)allocating the compute and network resources. It executes periodic checks in order to monitor the current availability of compute and network resources, and ensures that the NFVI adapts to traffic variations. Overall, the VNF lifecycle consists of the following:

- Day-0 configuration that includes VNF onboarding and resource allocation along with network service configuration.
- Scale-out, where horizontal scale-out involves creating more instances of a given VNF, for load balancing purposes and is typically triggered when the allocated CPU, memory or network resource utilization is increased upon increased traffic.
- Scale-in, which is the opposite process of scale-out and is triggered when a VNF is underutilized.
- Live migration that involves moving a VNF to a different hypervisor for optimization purposes, without service interruption [27]. It includes running both instances (in the old and new hypervisor) in parallel, while service migration is performed, and only migrating RAM contents as a final step.

4.3.4 System model

In our system model (Fig. 4.3), we focus on the VNF placement and resource allocation between the edge and the core tiers. At the core tier, there are M cloud hypervisors ($Cloud\{M\}$), with maximum capacity $HCloud_{Max}\{M\}$ and current utilization $HCloud\{M\}$ per hypervisor. Respectively, at the edge tier there are N MEC hypervisors ($MEC\{N\}$), with maximum capacity $HMEC_{Max}\{N\}$ and current utilization $HMEC\{N\}$ per hypervisor. We focus on the interconnection of each MEC hypervisor, in a leaf-spine topology with the cloud hypervisors. There are incoming VNFs in the system, some of which could be chained. For each $VNF_i\{Type, Resources, Hypervisor\}$ we define a type that is $HP\ LCVNF$, $LP\ LCVNF$ or $LTVNF$, the required resources that cannot exceed the hypervisor with the maximum capacity, and the hypervisor where it can be deployed. With respect to the service onboarding, we consider the following setup: i) the real-time applications, hosted in $HP\ LCVNFs$, are deployed to the edge, ii) the near real-time applications, hosted in $LP\ LCVNFs$ can be either allocated to the edge or core tiers, and iii) the non-real time applications, hosted in $LTVNFs$, are deployed to the core tier.

The scaling functionality of our system is being triggered based on the incoming requests per VNF, as multiple users can request data from the same VNF, resulting in increased VNF load. More specifically, we define the: i) scale-out threshold, which defines the value of the CPU utilization, above of which a new VNF of the same type is instantiated, ii) scale-in threshold, which defines the value of the CPU utilization, below of which the last created VNF is deleted, and iii) cooldown period, which is the predefined time interval that should pass before a consecutive scaling event at the same VNF may occur. Finally, the live migration functionality can be triggered upon a scale-in or scale-out event and involves only the shifting of the $LP\ LCVNFs$.

4.4 VNF orchestration algorithms

In this section, we discuss the role of the NFVO in the VNF lifecycle management, as well as the actual orchestration algorithms. In order to keep up in with the challenging cloud-native environments, where sub-second reaction times are sometimes required, and fast online algorithms are proposed. More specifically, the

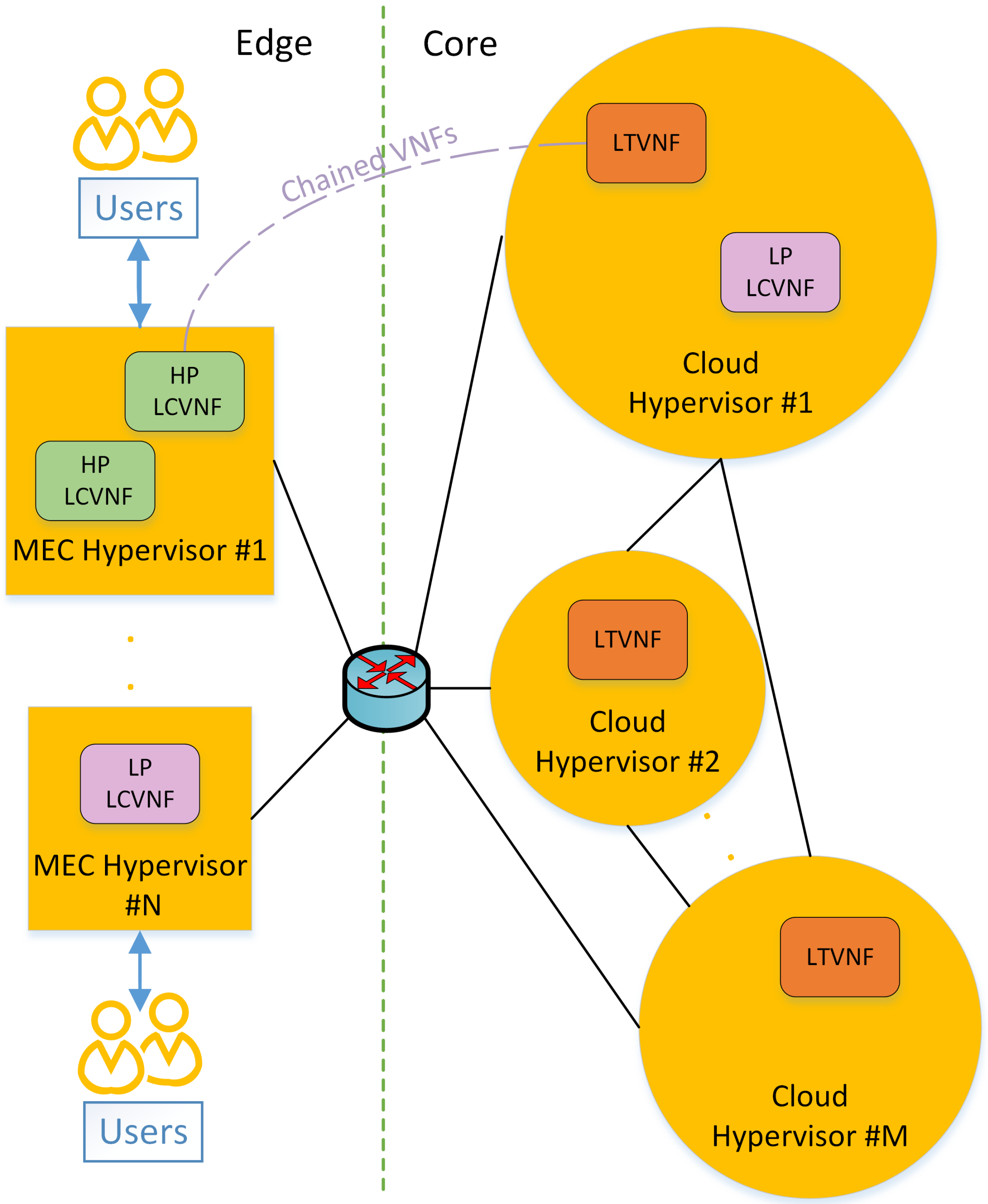

Figure 4.3: System model

VNF scheduling problem is split in three phases, which are centrally controlled by the NFVO:

- The VNFFG embedding phase is executed once during service initialization and onboarding, to allocate VNFs to the MEC or cloud hypervisors, based on

Algorithm 1 VNFFG embedding

```
1: The VNFFG embedding process starts from the services
   with the highest QoS. We traverse all VNFs in the VNFFG
   breadth-first, starting from the entry point where the UE
   connects.
2: If the latency constraints of the VNF links exceed the
   round-trip time to the cloud, the VNF is assigned to the
   MEC. Otherwise, it is assigned to the cloud.
3: If the MEC resources are exhausted, further deployment
   of $HP\ LCVNFs$ is blocked, as well as their chained VNFs,
   unless they can tolerate the increased latency associated
   with the core tier ($LP\ LCVNFs$).    =0
```

delay constraints.

- The service scale-out is performed periodically, based on a user-defined cooldown period, and triggers a scheduling operation for all scaled-out VNFs. A fast online algorithm is devised to handle this operation, while a live migration step might be performed in cases of insufficient edge resources.
- The service scale-in is also a periodic process, which erases VNF instances when the user demand decreases, to free up resources when they are not needed. We propose a live service migration step to be performed after the scale-in operation to further optimize the VNF placement.

VNF scheduling is based on a cost function, which takes into account the hypervisor resources consumed by the VNF (i.e., CPU, memory and disk size), as well as bandwidth costs to interconnect the VNFs in the VNFFG. In general, the minimum cost is achieved when all VNFs of the same VNFFG are placed at the same hypervisor. It gradually increases as VNFs are placed at different hypervisors occupying network links for communication, while MEC hypervisors are generally assigned a higher cost than cloud hypervisors.

4.4.1 VNFFG embedding

Although many different topologies have been considered in the literature for the core and edge tiers, in this work we consider a standard leaf-spine topology. This topology simplifies the VNFFG routing over the physical infrastructure, as all hypervisors in the core and edge tier are interconnected in a mesh with a fixed number of hops (Fig. 4.3). The VNFFG embedding is performed during the service bootstrapping phase at the NFVO that assigns VNFs to the core or edge tier based on their delay constraints. The edge tier hosts have a higher OPEX cost than the core tier hypervisors and hence a higher deployment cost which is reflected on the cost function. Thus, typically only a limited number of edge VNFs is deployed to the edge. Algorithm 1 explains the basic steps of the VNFFG embedding process.

4.4.2 Online VNF scheduling

VNF scheduling is an online problem, as VNFs are typically scaled-out and scaled-in within very fast time-frames, in the order of seconds, based on current traffic. Although many works in the literature solve an offline version of the problem, where the total number of VNFs is known during service bootstrapping, and do not take into consideration the real traffic of the VNFs, this assumption is not valid in modern cloud infrastructures. In this work, we assume that only the VNF assignment to the core or edge tier has been completed during the service bootstrapping phase, hence the online scheduling algorithm only needs to assign the VNF to the actual cloud or MEC hypervisor. In what follows, VNFs are placed in hypervisors with sufficient compute, memory and networking resources. This algorithm tries to first accommodate the highest cost VNFs, starting from the hosts with the highest available resources. The main algorithmic steps of the proposed Alg. 2 for scheduling scaled-out VNFs are explained as follows and they are generally performed after a predefined cooldown period has elapsed. Furthermore, the algorithm tries to accommodate higher priority VNFs via live migration actions of lower priority VNFs, while it tries to restore the balance of the system after a scale-in process. Please note that our algorithm can be executed in combination with any NFVO that supports scaling capabilities and VIM with live migration support.

In more detail, regarding the scale-out operation, we try to place the new VNF at the same hypervisor with the original VNF that is being scaled, in order to eliminate inter-hypervisor delays. Table 4.1 depicts the actions that are being performed, depending on the triggering event. As Fig. 4.4 demonstrates, in case the original VNF resides in a MEC hypervisor and there are available resources, the new VNF is allocated to the same hypervisor as well. In case of insufficient

Algorithm 2 Online VNF scale-out/scale-in and dynamic live-migration scheduling.

```
    Input: $HMEC_{Max}\{N\}$, $HCloud_{Max}\{M\}$,
    $HMEC\{N\}$, $HCloud\{M\}$,
    $VNF_i\{Type, Resources, Hypervisor\}$
    Triggering Event $e$, where $e$ in $\{scale-in, scale-out\}$
    $VNF_e$
    Output: $Hypervisor$ for VNF placement
 1: if $e = scale-out$ of $VNF_e$ then
 2:   if $VNF_e\{3\} = MEC\{e\}$ then
 3:     repeat
 4:       if $VNF_e\{2\} \leq MEC\{e\}$ then
 5:         allocate $VNF_e$ on $MEC\{e\}$
 6:         update $MEC\{e\}$ resources
 7:       else if $VNF_e\{1\}$=$LP\ LCVNF$ AND $VNF_e\{2\} \leq max(HCloud)$ then
 8:         allocate $VNF_e$ on $max(HCloud)$ & flag it
 9:         update max(HCloud)
10:       else if $VNF_i\{1\}$=$LP\ LCVNF$ exists on $MEC\{e\}$ then
11:         if $max(VNF_i\{2\}) \leq max(HCloud)$ then
12:           live migrate $VNF_i$ to $max(HCloud)$ & flag it
13:           update $HMEC\{e\}$
14:           update $max(HCloud)$
15:         end if
16:       else
17:         reject scale-out request
18:         exit algorithm
19:       end if
20:     until $VNF_e$ is allocated
21:   else if $VNF_e\{3\} = Cloud\{e\}$ then
22:     if $VNF_e\{2\} \leq HCloud\{e\}$ then
23:       allocate $VNF_e$ on $Cloud\{e\}$
24:       update $HCloud\{e\}$ resources
25:     else if $VNF_e\{2\} \leq max(HCloud)$ then
26:       allocate $VNF\{e\}$ on $max(HCloud)$
27:       update $max(HCloud)$ resources
28:     else
29:       reject scale-out request
30:       exit algorithm
31:     end if
32:   end if
33: else if $e = scale-in$ of $VNF_e$ then
34:   release $VNF_e\{2\}$
35:   update $hypervisor_e$ resources
36:   if $VNF_e\{3\} = MEC\{e\}$ then
37:     while flagged $LP\ LCVNF$ exist on Cloud AND $HMEC\{e\} \geq$ flagged
        $VNF_i\{2\}$ do
38:       live migrate flagged $VNF_i$ to $HMEC\{e\}$
39:       update $HMEC\{e\}, HCloud$
40:     end while
41:   end if
42: end if =0
```

Table 4.1: Triggering events and actions

	Action #1	**Action #2**	**Action #3**
Scale-out	Find VNF placement hypervisor without SLA violation. Perform migrations, if necessary.	Allocate hypervisor resources	Instantiate VNF
Scale-in	Terminate VNF	Release hypervisor resources	Migrate flagged VNFs

MEC resources, in the event of: i) $LP\ LCVNF$ type, it can be directly allocated to the cloud hypervisor with the most free resources, and is being flagged, ii) $HP\ LCVNF$ type, a live migration of existing $LP\ LCVNFs$ takes place to the cloud hypervisor with the maximum available resources, starting with the VNF that occupies the most resources, in order to free up MEC resources for the incoming VNF, along with updating the bookkeeping of the migrated VNFs (flagging), or iii) no $LP\ LCVNF$ type exists at the MEC hypervisor, the scale-out request gets rejected. Conversely, on the occasion of a scale-in triggering event, the resources of its hypervisor are released. In case of a MEC hypervisor, we migrate back possible flagged $LP\ LCVNFs$, according to our bookkeeping.

Overall, the runtime complexity of the proposed algorithm is $O(n^2)$, as it is determined by the most important term, i.e., the $max()$, which is nested in one loop:

- Scan the VNF array to find $LP\ LCVNFs$ at the MEC hypervisor $\longrightarrow O(n)$
- Calculate the maximum value of the array $HCloud \longrightarrow O(n)$, nested in one loop $\longrightarrow O(n^2)$.

In terms of runtime memory, we need:

- Four one-dimensional arrays to store the maximum and current capacities of the MEC and cloud hypervisors. Specifically, two arrays of size N for the $HMEC_{Max}$ and $HMEC$ values, and two arrays of size M for the $HCloud_{Max}$ and $HCloud$ parameters will be allocated to the runtime.
- One two-dimensional dynamic array with size $i{\times}3$, to store the $Type$, $Resources$ and $Hypervisor$ data for each VNF_i.

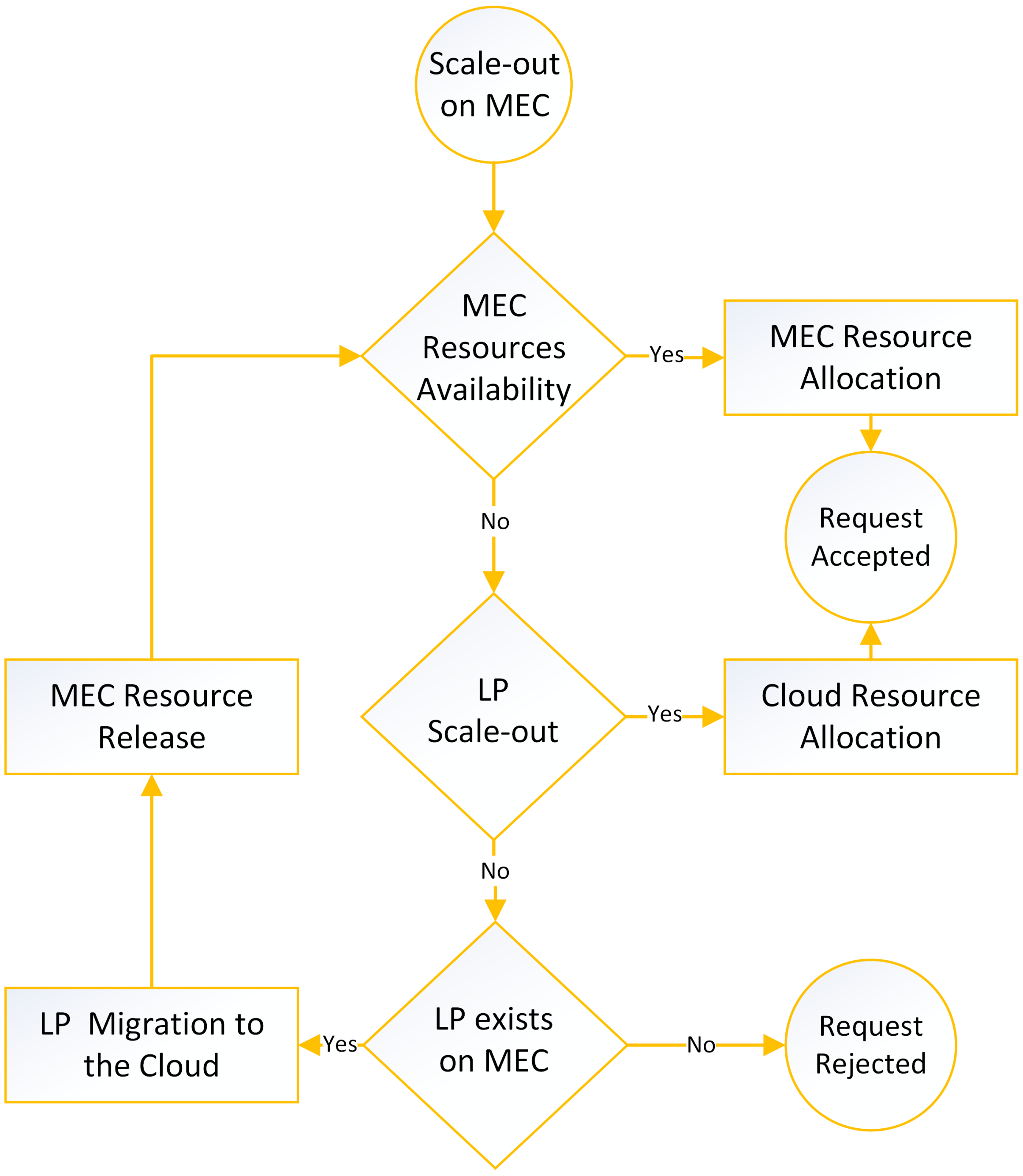

Figure 4.4: Flow chart for scale-out triggering event of LCVNF on MEC

Regarding the execution of Alg. 2, assuming the presence of i VNFs in the system, the maximum number of iterations can be calculated for the worst-case scenarios. Specifically, for the first loop of the algorithm (steps 3-20), the maximum number of iterations is $(i-1)$. This occurs when a $HP\ LCVNF$ needs to be scaled-out and all the remaining VNFs at the MEC are $LP\ LCVNFs$ and must be all be migrated to the cloud to release sufficient resources for the high priority function.

Table 4.2: Hardware characteristics

	Controller Node	Cloud Compute Node x2	MEC Compute Node	OSM
CPU	Intel i5-8500	i5-8500	i5-7400	i5-8400
Cores	6	6	4	6
RAM	32GB	16GB	8GB	8GB
Storage (SSD)	250 GB x2	250 GB	120 GB	120 GB
NIC	2x1Gbps ETH	2x1Gbps ETH	2x1Gbps ETH	2x1Gbps ETH

Conversely, the maximum number of iterations for the second loop of Alg. 2 (steps 37-40) is also $(i-1)$, under the occasion that the aforementioned $LP\ LCVNFs$ return to their original hypervisor after the scale-in process of the $HP\ LCVNF$.

4.5 Testbed implementation

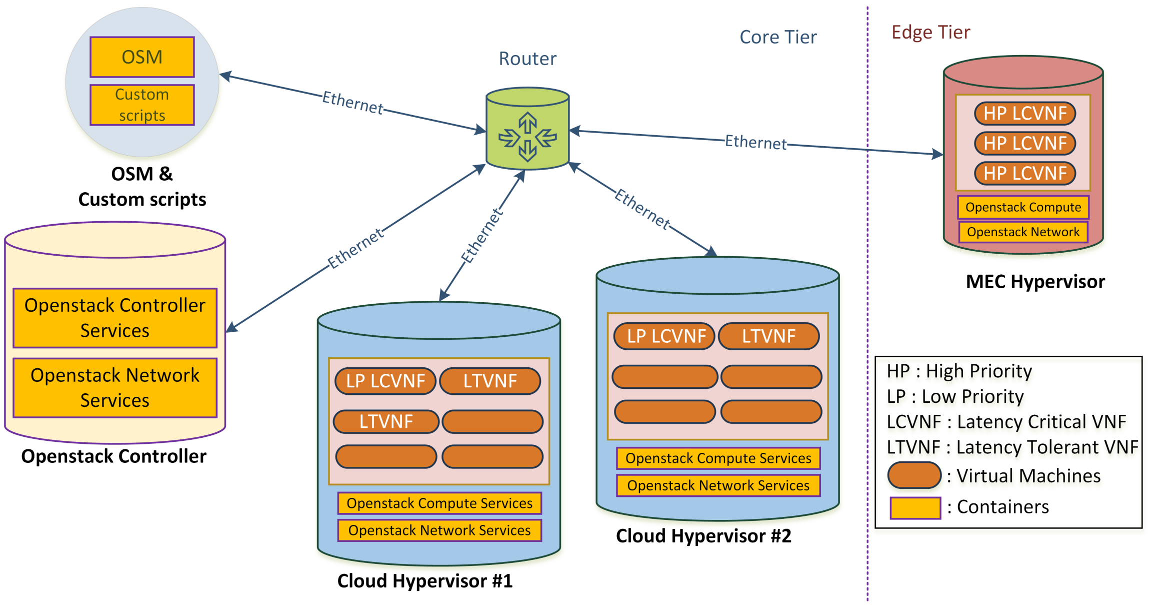

Figure 4.5: MEC-enabled 5G testbed

In order to demonstrate the potential of the described architecture, we introduce a real implementation of a MEC-enabled 5G testbed, depicted in Fig. 4.5. The hardware of the testbed, as seen in Table 4.2, consists of five physical servers, where the functionalities of the core tier (e.g., the cloud and the NFVO) and the edge tier (e.g., the MEC) are deployed to, as well as another physical server that enables the management, in terms of infrastructure virtualization. In terms of compute resources, the physical server at the edge site has significantly lower computational power compared to the servers at the core. In terms of networking, the physical servers are connected to two routers through 1 Gbps ETH interfaces.

With respect to the software installation, openstack, on its "Queens" release, is the open-source platform that acts as the VIM, in order to deploy and control the VMs that will host the VNFs. The openstack controller node, deployed to one physical server as shown in Fig. 4.5, hosts the compute and network management components for the virtualization and management of the infrastructure, while the compute nodes (or hypervisors), deployed to three physical servers, provide a pool of physical resources, where the VMs are being executed. Openstack is based on services, and in order to provide the needed isolation and management, they are deployed to LXD containers [75]. For instance, the nova service, part of the openstack compute services that reside in all compute nodes, is responsible for spawning, scheduling and decommissioning the VMs on demand, while the neutron service, which resides in all four nodes, is responsible for enabling the networking connectivity. Additionally, the openstack telemetry service (based on the ceilometer service) is deployed to collect monitoring data, including system and network resource utilization, based on which further actions are taken. All nodes need two network interfaces, namely, the management (i.e., control plane), for the communication among the openstack services and the NFVO, and the provider network (i.e., data plane), for the communication among the VMs, while each application has its own virtual tenant network.

Table 4.3: Experimental setup

Parameter	Value	Parameter	Value
$HMEC_{Max}\{1\}$	3 vCPUs	$HCloud_{Max}\{1\}$	6 vCPUs
HP LCVNF max latency	100ms	LP LCVNF max latency	200ms
LTVNF max latency	Irrelevant	Resources per VNF	1 vCPU
Hypervisor HP LCVNF	MEC	Hypervisor LP LCVNF	MEC/Cloud
Hypervisor LTVNF	Cloud	Cooldown Period	180s
Scale-out	90% CPU utilization	Scale-in	30% CPU utilization

Moreover, it is worth noting that openstack supports two important features, namely the horizontal scaling that is the expansion of the physical resources, simply by adding new physical servers where the compute node services are deployed to, and the live migration of the VMs. The migration is classified as live due to the fact that after a VM migration is complete, the VM status resumes exactly from the same state it was before the migration, without service interruption. The duration of the live migration might take from few seconds to several minutes, depending on various factors, including, but not limited to: i) the virtualization platform, ii) the underlying hardware, iii) the type of the hypervisor, iv) the type of storage, v) the footprint of the VMs in terms of vCPUs, vRAM and storage, vi) the current network load, and vii) the current VM load. Without any doubt, there should be an upper limit for the duration of the live migration, in order for the system to be agile and adaptive to the real-time traffic changes, but this is related to the actual system and the limitations that are being imposed by the hardware, the architecture decisions and the virtualization platform.

The NFVO, which is responsible for the computing and network resource orchestration and management, is deployed as an independent entity to the fifth server, at the core tier, in alliance with the ETSI NFV information models, and is based on the OSM, on its sixth release. Although there is a variety of NFVOs in the literature [67], its low hardware requirements, combined with the capabilities it offers, made OSM the most suitable NFVO for our system. OSM supports descriptor files written in yaml, namely the VNFD and the NSD. The former defines the needed VNF resources in terms of compute resources and logical network connection points, the image that will be launched at the VM, as well as the auto-scale thresholds (e.g., scale-in, scale-out and cooldown period, minimum or maximum number of VNFs) based on the metrics that are being collected from the telemetry service of the VIM. The latter is responsible for the connection point links, using virtual links, among the interconnected VNFs, mapping them to the physical networks provided by the VIM.

Neither openstack nor OSM are aware of the type of service that is being executed at the VNF. Furthermore, OSM is not aware of the hypervisor where the VM that hosts the VNF is being placed and leaves the VM placement to openstack. This lack of placement control deprives OSM of controlling the migration feature. In order to gain control of this feature, we implemented this functionality with a bash script. Openstack supports four different placement methods via the compute schedulers; filter scheduling, based on filters and weights, chance scheduling that randomly selects the compute filters, utilization aware scheduling, based on actual resource utilization and availability zones scheduling where the compute nodes are divided into zones. None of the above options, though, take into account the actual service running at the VM or how to allocate $LCVNFs$ or $LTVNFs$ among the hypervisors. To that end, we devised the aforementioned bash script, which is based on our two proposed algorithms, and performs the onboarding, scale-out/in and live migration actions of the VNFs to the appropriate hypervisors.

4.6 Experimental results

In order to demonstrate the potential of the described architecture, we conducted a set of experiments, leveraging the MEC-enabled 5G testbed, as described in section 4.4. In the following, we first provide an experimental setup and, then, we evaluate the performance of the proposed algorithms.

4.6.1 Experimental setup

In our testbed setup, we assume one MEC and one cloud hypervisor. For our experiments, we define the max latency as: i) $100ms$ for the $HP\ LCVNF$, and ii) $200ms$ for the $LP\ LCVNF$. For the $LTVNF$, the latency is irrelevant as the transmission is asynchronous. Note that the latency is measured as the E2E delay between the UE and the hypervisor, also corresponding to the response time of the application. The scale-out threshold is set at 90% CPU utilization, the scale-in at 30% and the cooldown period at $180s$. Since we assume exponential service time of the $LCVNF$ service, as soon as the CPU utilization exceeds the predefined threshold, the response time violates the SLA, so the scale-out process will take place prior to this violation. Please note that the aforementioned values are fully customizable, depending on the actual requirements of the diverse 5G applications. Table 4.3 depicts the experimental setup in detail. Finally, the following three experiments run multiple times, separately, in a duration of 24 hours each.

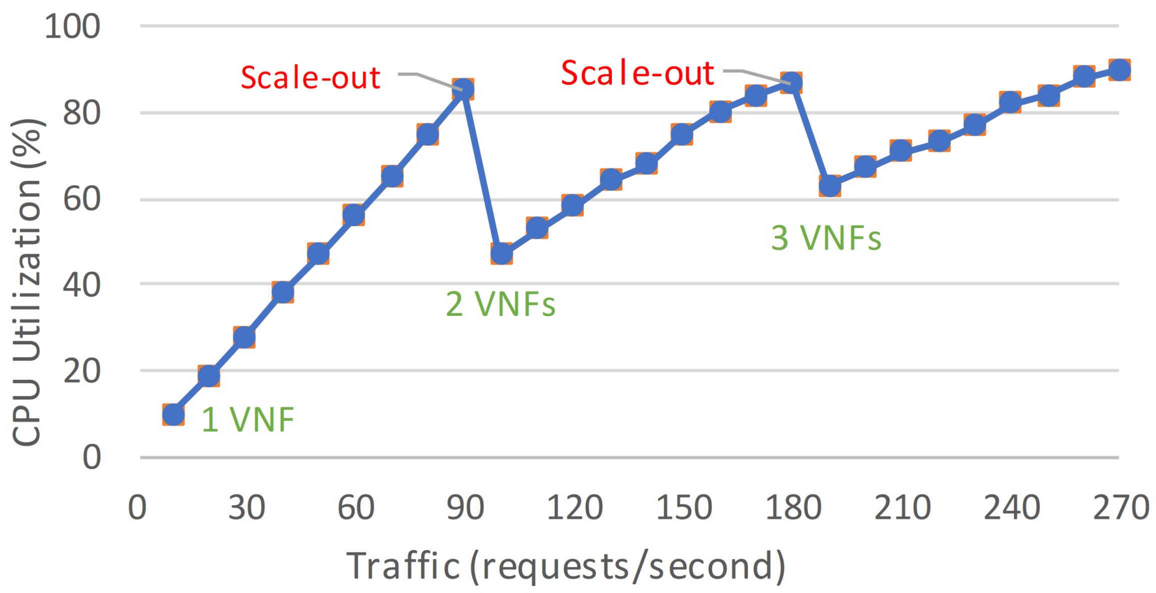

Figure 4.6: Scale-out process to accommodate increased incoming traffic

Since most of the parameters were deterministic, the results were stable, with a variation of 2ms.

4.6.2 Autoscaling experiment

In the first experiment, Fig. 4.6, we demonstrate the scale-out process. We start with one $HP\ LCVNF$ and, as the traffic increases, the CPU utilization of the VNF increases accordingly. When it reaches the CPU utilization threshold at 90%, it is scaled-out and a second $HP\ LCVNF$ is being instantiated. In order to equally distribute the traffic between the two VNFs, we deploy a load balancer VM with a round robin balancing policy. Hence, each VNF has approximately 45% CPU utilization when the new VNF is instantiated. While the traffic is further increased, another scale-out event is triggered and a third $HP\ LCVNF$ is instantiated, with the load balancer distributing the incoming requests to three VNFs. This results in a 60% CPU utilization by the time the third VNF is instantiated. The measured scale-out duration for a VM with 1 vCPU, 1GB of RAM and 5GB of storage, from the moment of the initial command to the VIM until the instantiation process was complete, was 15 seconds.

As we increase the traffic over time, we observe that the angle that is formed between the x axis and the graph on Fig. 4.6 is reduced, according to the number of VNFs in the system that serve the requests. This is expected, as the traffic is equally

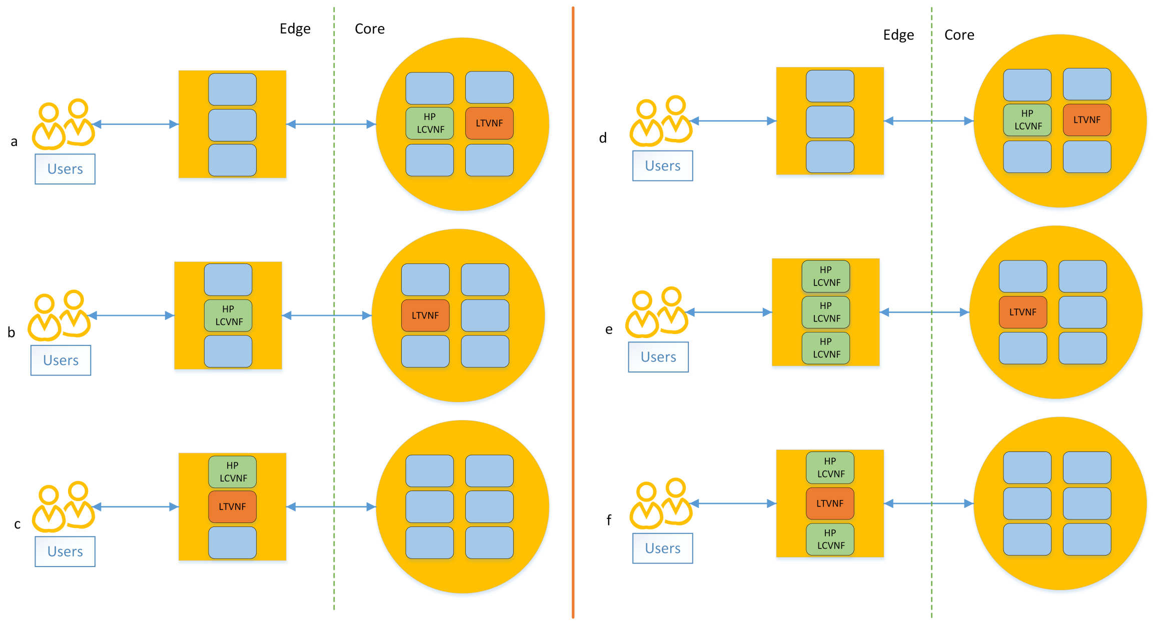

Figure 4.7: VNF Initial Placement (left) and placement after auto-scale functionality (right)

distributed to two or three VNFs, while the traffic rate is increasing in a steady pace. With the autoscaling feature, we can accommodate more requests, compared with the legacy monolithic deployments that do not support such feature.

4.6.3 Embedding algorithm & placement experiment

In the second experiment, illustrated in Fig. 4.7, we demonstrate the various placement locations, validating that the algorithm for the onboarding process (i.e., Alg. 1), provides the optimal placement result for maximizing the served requests. More specifically, we assume two chained VNFs, one $HP\ LCVNF$ and one $LTVNF$, and we investigate three possible VNF placement methods: i) all VNFs deployed to the cloud (Fig. 4.7-a), ii) the $HP\ LCVNFs$ deployed to the MEC and the $LTVNFs$ to the cloud (Fig. 4.7-b), and iii) all VNFs deployed to the MEC (Fig. 4.7-c). We reject the first solution as the SLA is being violated, because the $HP\ LCVNF$ cannot tolerate the increased latency imposed by the MEC-cloud link. According to the embedding algorithm, the initial placement is performed based on latency constraints, where the $HP\ LCVNFs$ are allocated to the edge tier and the $LTVNFs$ are allocated to the core tier. After the initial placement, the $HP\ LCVNF$ is hosted in the MEC ($VNF_1\{HP, 1, MEC\}$), while the $LTVNF$ is hosted in the cloud ($VNF_2\{LTVNF, 1, Cloud\}$). This is the optimal placement solution as, in case of increased traffic, the $HP\ LCVNF$ can scale-out twice, until the MEC resources are depleted ($HMEC = 0$), and serve more requests (Fig. 4.7-e). Finally, in the third deployment method, where everything is deployed to the MEC, the $HP\ LCVNF$ can scale-out only once (Fig. 4.7-f) and the MEC resources are depleted ($HMEC = 0$), since there is one $LTVNF$ deployed to the MEC ($VNF_2\{LTVNF, 1, MEC\}$) that occupies 1 vCPU.

In Fig. 4.8, the response time of the VNFs versus the traffic is depicted, depending on the hypervisor that the VNFs are placed. From this figure, we can observe that if all the VNFs are deployed to the cloud, no further investigation is performed as this deployment method violates the SLA (over 100ms) for the $HP\ LCVNF$. For the MEC-cloud placement method where VNFs are placed between the MEC and the cloud, the system will be able to support up to three $HP\ LCVNFs$ at the MEC in order to serve up to 270 requests/s without violation of the SLA. Finally, while the third deployment method has improved response time due to the elimination of the link for the communication of the $HP\ LCVNF$ with the $LTVNF$ (they are hosted in the same hypervisor), the total requests/s that can serve are limited up to 180, due to the fact that the MEC resources quota has been reached. In conclusion, we notice a 50% increase of accepted latency critical incoming requests when both edge and core resources are employed.

4.6.4 Online VNF scheduling experiment

In the third experiment, depicted in Fig. 4.9, we demonstrate how the live migration feature can be employed to support more requests when $LCVNFs$ with different priorities are competing for the same MEC resources, without interrupting the availability of the near real-time application. In this scenario, we take advantage of the live migration feature, described in Alg.2. Initially, the embedding algorithm allocates both VNFs to the edge tier (Fig. 4.9-a) ($VNF_1\{HP, 1, MEC\}$, $VNF_2\{LP, 1, MEC\}$). While the requests for the VNF_1 are increasing, the CPU utilization increases as well, resulting in the scale-out of VNF_1 ($VNF_3\{HP, 1, MEC\}$). When a second scale-out ($VNF_4\{HP, 1, MEC\}$) takes place, the MEC resources have been depleted ($HMEC = 0$), triggering the scheduling algorithm to: i) live migrate the $LP\ LCVNF$ to the cloud ($VNF_2\{LP, 1, Cloud\}$), as depicted in 4.9-b, and ii) place the scaled-out $HP\ LCVNF$ (VNF_4) at the MEC (Fig. 4.9-c). When

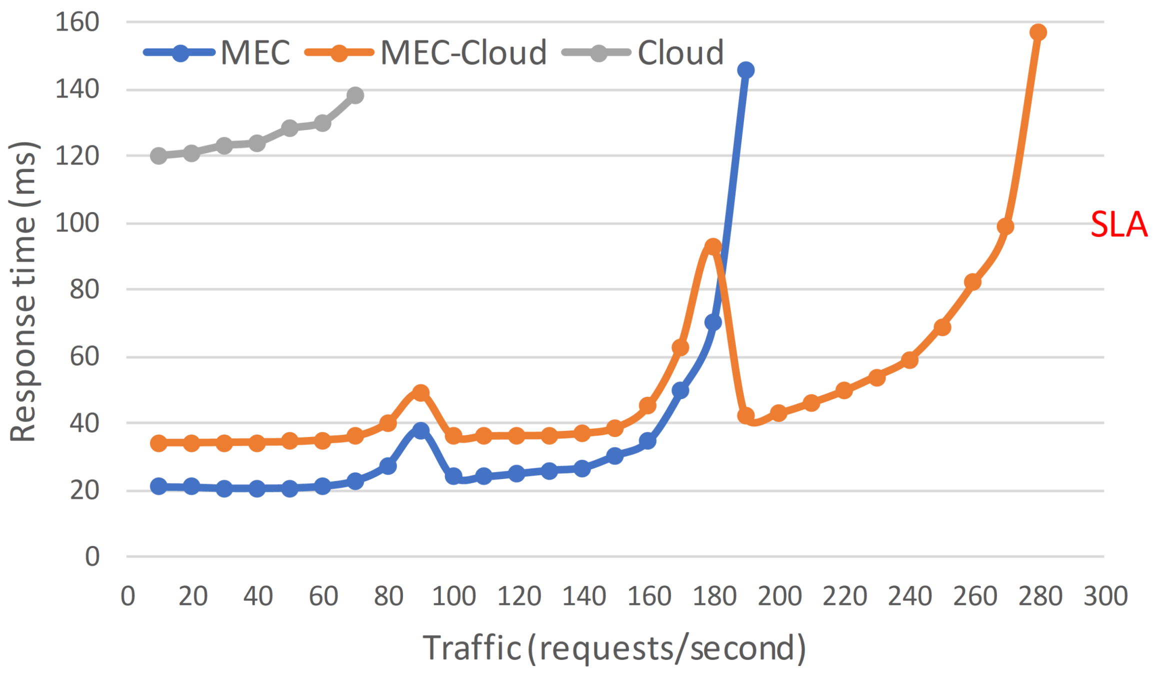

Figure 4.8: Response time over traffic for the different deployment scenarios

the traffic at the $HP\ LCVNF$ is decreased, a scale-in (termination of VNF_4) occurs and the $LP\ LCVNF$ (VNF_2) is migrated back to its original hypervisor (Fig. 4.9-d).

In Fig. 4.10, we evaluate the response time versus the time in minutes. The requests for the $HP\ LCVNF$ are increased over time while the requests for the $LP\ LCVNF$ are stable. As the $HP\ LCVNF$ needs to scale-out at the minute 85, the script commands the VIM to live migrate the $LP\ LCVNF$ from the MEC to the cloud, thus freeing up resources for the scale-out of the $HP\ LCVNF$. The live migration process, at the minute 85, lasts 28 seconds, for a VM with 1 vCPU, 512MB RAM and 3GB local storage, while no service interruption was observed. Please note that during the live migration process, we notice a slightly increased response time for the $LP\ LCVNF$ that is not violating the SLA neither during nor after the migration has been completed. Finally, when the scale-in action occurs at the minute 145, the $LP\ LCVNF$ is migrated back to its original hypervisor, in accordance with Alg. 2.

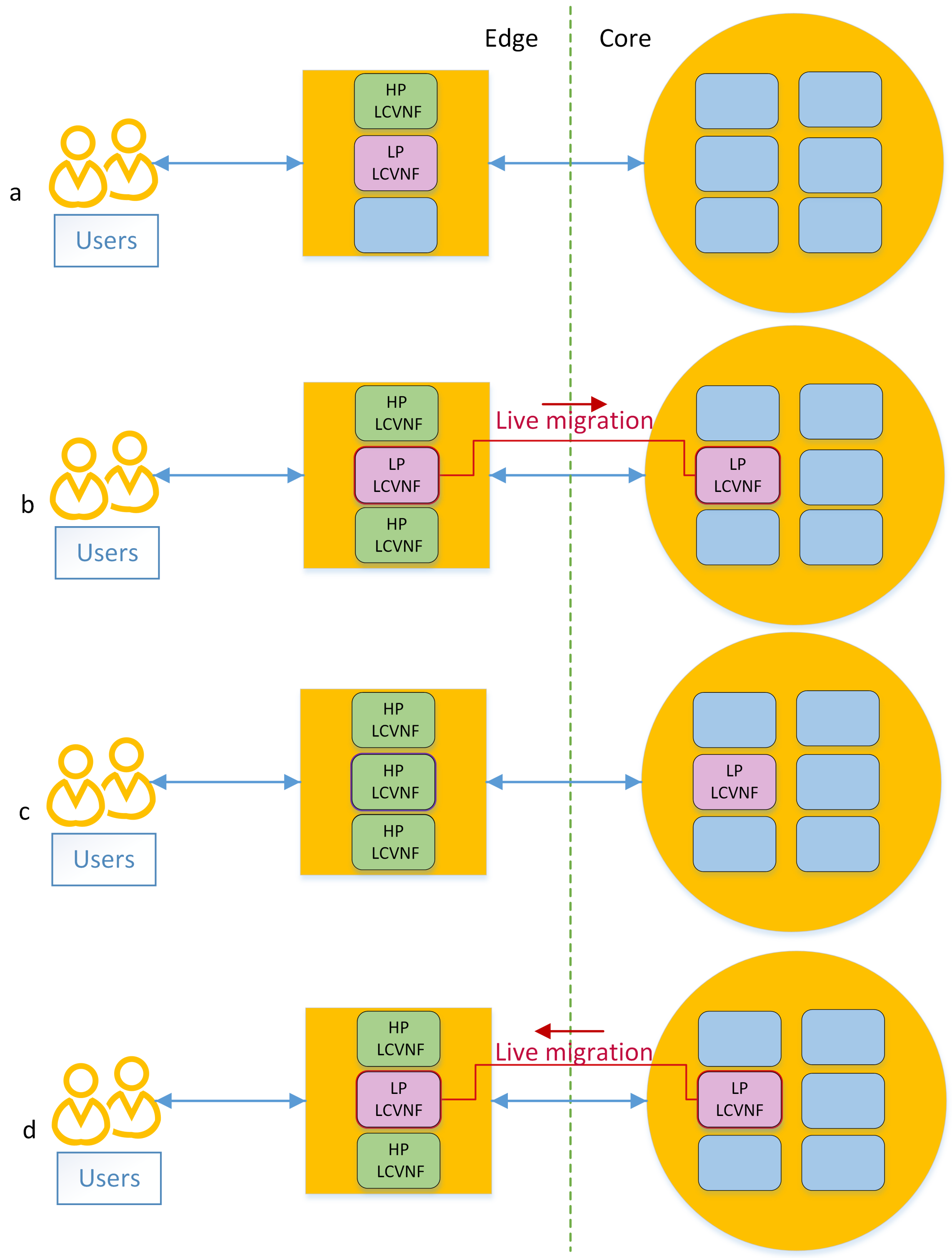

Figure 4.9: Live migration to accommodate more HP LCVNF at the edge

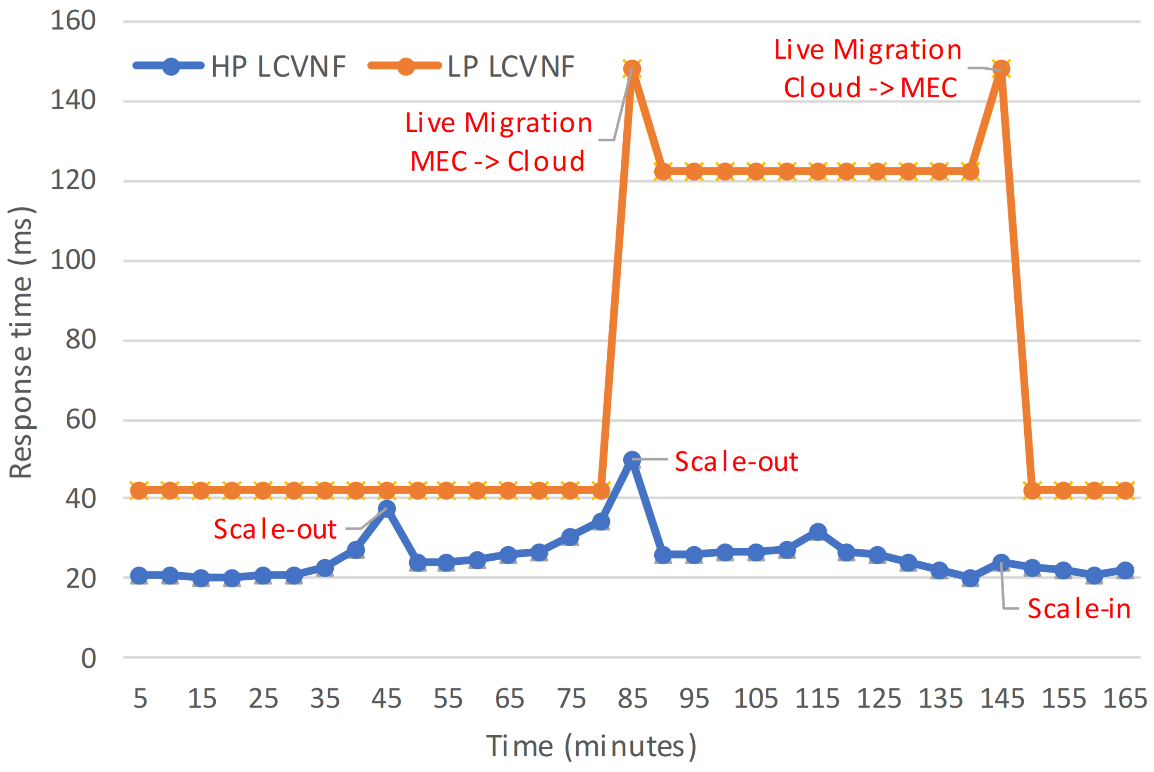

Figure 4.10: Response time pre and post migration

4.7 Conclusion

In this chapter we presented a MEC-enabled 5G IoT architecture, able to exploit the interplay between the core and edge tiers in an NFV environment. We discussed the key enabling technologies and filled the gap between the NFVO and the VIM entities by proposing embedding and scheduling algorithms for the initial placement and online reallocation of the VNFs respectively, leading in enhanced VNF LCM. We applied our algorithms to a fully deployed MEC-enabled 5G testbed implementation, where applications with different priorities and latency constraints have been executed. The conducted experiments shown that through the proposed schemes, a better utilization of MEC and cloud resources can be obtained on the fly, enabling the system to serve a higher number of latency critical applications without SLA violation.

Chapter 5

Cost-aware placement and enhanced lifecycle management of service function chains in a multi-domain 5G architecture

5.1 Introduction

Novel technologies, such as SDN and NFV, have shifted the legacy monolithic wireless telecommunication system towards a traffic-adaptive and service-aware network, leading in the realization of the 5G architecture. On the one hand, NSA implementations have appeared in recent years, taking advantage of the current LTE infrastructure that MNOs already possess, bringing 5G capabilities to the LTE network [30], where a virtualized EPC van be combined with the 4G LTE or the 5G NR RAN. On the other hand, innovation with actual ultra-low latency and high bandwidth support can be optimally achieved by the SA framework, where only 5GC NFs and 5G NR RAN are implemented [28].

5G use cases might require ultra-low E2E latency (i.e., under 10ms, or even under 1ms), extremely high availability, (i.e. over 99.999%), high bandwidth (i.e., up to 10 Gbps), or a vast amount of loosely connected devices, according to the IoT paradigm [76, 77]. LTE networks were not designed for the demanding 5G-enabled use cases, thus, even with NSA 5G implementations, 5G use cases cannot run at their full extent. Vertical markets, such as automotive, entertainment, industry 4.0 and manufacturing, healthcare and many more, will create an innovative ecosystem to provide their services, exploiting the 5G capabilities[78]. The ability to bring networking and processing power to the edge (i.e., to the end user proximity), enabled by MEC architectures [79], further enables the 5G use case (i.e., services provided by 5G verticals) realization. MEC architectures support both decentralized (i.e., edge) and centralized (i.e., core) network and compute resource capabilities, taking advantage of the established cloud computing paradigm across the whole extent of the infrastructure, a crucial requirement towards the low latency requirements of selected 5G vertical services [80]. Finally, by supporting a leaf-spine networking topology [81], the network can support stable and known values of latency across the network entities.

In order to support the 5G SA realization, investments and daily operational expenses need to be paid by the MNOs, the cost of which is divided in CAPEX and OPEX costs, respectively. The forecast for the automotive vertical, for C-V2X use cases, predicts that three billion euros in CAPEX will be invested by 2030 in the EU [11]. Yet, it is not expected to provide complete coverage until 2035, where the prediction for full population coverage is calculated to 4.8 billion euros. Moreover, this analysis emphasizes to the need of synergies, where MNOs collaborate and share their infrastructure in order to minimize the deployment and investment cost, saving thus 275 million euros. Another analysis for the entertainment vertical, and specifically for broadcasting use cases, expects a 730 million euros in CAPEX investment, that will cover roughly the one third of the population of the EU by 2030, while the OPEX cost forecast reaches the amount of 214 million euros [12]. In this report, it is estimated that the coverage of 20% of the population living in rural areas costs about half the entire running cost of the network for the whole country, enhancing thus the cost benefits of a shared infrastructure scheme.

5G verticals need a seamless infrastructure that is able to support a vast amount of information, high throughput and low latency per case, wide area coverage that could extend from the premises of a factory, up to different countries, and handover capabilities, where mobility exists [82]. The infrastructure capabilities of a single MNO, or a federation of MNOs with LCM functionalities, cannot meet the 5G vertical requirements. Thus, a cross-operator compound with multi-domain support (i.e., federation of providers) along with a full LCM assistance scheme, seems to be the most viable solution. This paradigm that has recently appeared, requires the cooperation across MNOs in order to provide a unified underlying infrastructure for the smooth execution of the 5G verticals. Additionally, fully leveraging the NFV capabilities, 5G vertical applications can be deployed in a sequence of multiple VNFs, resulting in the formation of SFCs. Depending on the use case, SFCs may be required to be deployed to the end user proximity, wherever the user may be located, a condition that only a unified infrastructure can support. The underlying infrastructure may belong to the MNO (i.e., local) or to a partner MNO (i.e., foreign). While there is a different deployment and execution cost on each environment, SFCs can be deployed uninterrupted as the infrastructure appears unified towards them.

The SFC placement problem (i.e., the location selection where the building blocks of the SFCs are deployed) has been studied under various objectives, including, but not limited to, the QoS, the resource usage, the energy consumption, the cost and the reliability [27]. Additionally, the SFC LCM capabilities are handled by the NFVO [4], which is responsible for managing and orchestrating the underlying federated resources by issuing commands to the VIM, respecting, at the same time, the 5G vertical application's SLA. With the aid of LCM functionalities, such as migration and scaling, the orchestration of the SFCs can be further optimized, having an impact at the service delivery and the overall deployment cost.

While there is a variety of recent works on VNF placement within the premises of one single domain [20, 52, 83–89], there are less works in multi-domain environments [21–23, 90, 91], and very few relevant works on the SFC orchestration [23, 85, 89]. With respect to the architecture, most of the works focus on the VNF placement problem where the resources are only centralized ([20, 22, 83–86]), and ignore the placement at core-edge environments ([52, 89]). Moreover, the LCM functions of migration and scaling are being considered in [52, 85, 89], while authors in [87] and in [23] consider only the scaling and migration LCM functions, respectively. Finally, with reference to the experimental results, all the aforementioned works provide simulation based results, except authors in [21, 52, 89] that provide an implemented testing environment for further verification. The simulations can provide approximations, while a real-life system can execute actual tasks, thus providing invaluable results.

In this chapter, we provide the following contributions. By utilizing cutting edge networking technology concepts with widely accepted and applied VTs, we employ a novel leaf-spine multi-domain NFV-enabled architecture. The architecture is spread across core and edge infrastructure that utilizes both local and foreign domains for the 5G verticals to be uninterruptedly executed, respecting, at the same time, the required SLAs for their applications that are implemented as SFCs. With respect to the QoS, in terms of latency and compute resources, we propose an online cost-aware ILP formulation, constructing the SFC placement problem as a ILP optimization problem, solving it using the CPLEX [92] solver to provide the optimal solution based on the cost per minute (CPM) of the SFC. Furthermore, we propose a less complex online hop-based heuristic algorithm that finds a near-optimal solution for the SFC placement problem. We further support the initial placement by adding the enhanced NFV-based LCM techniques of live migration and scaling that get triggered on-the-fly, based on real traffic patterns, and lead to a cost-efficient usage of the local and foreign resources. Finally, we provide a comprehensive variety of experiments, based on our custom built openstack-based [37] 5G experimental platform. While the previous chapters focus on the LCM and VNF scheduling of architectures that support IoT use cases, the current chapter provides the SFC placement intelligence and further extends the LCM capabilities in a general purpose NFV architecture.

The remainder of this chapter is organized as follows. Section 5.2 presents in detail the related works. Section 5.3 provides the proposed multi-domain 5G architecture, along with the relevant NFV key concepts, the system model and the cost interpretation. Section 5.4 discusses the proposed algorithms for the solution of the SFC placement problem. Section 5.5 delivers the LCM functionalities of the live migration and scaling. Section 5.6 explains the 5G experimental platform implementation and the employed open-source tools for its realization. Section 5.7 analyzes the obtained experimental results, thoroughly explaining the different experimental scenarios, whereas section 5.8 is devoted to the chapters conclusions.

5.2 Related work

While there are several works on the VNF placement in a single domain [20, 52, 83–87], there are very few works that tackle both orchestration and LCM techniques in an NFV-based multi-domain environment, where multiple operators cooperate in order to support the 5G vertical services and applications [21–23]. From the aforementioned works, [21, 87] are tackling the issue from a cost-aware scope when it comes to core-edge environments, where pricing policies and resource availability may greatly vary.

5.2.1 Single-domain VNF placement

Regarding the single domain works, authors in [83] propose a resource allocation strategy for energy-aware SFC in SDN-based networks. Although multiple heuristics are provided for different optimization problems, the authors consider a single administrative domain with a centralized datacenter. Considering the placement method, [52, 83, 84, 86, 89] are focusing on the QoS, while [85] on the cost. Authors in [20, 87] provide both a cost and QoS–aware placement method. In terms of network topology, only [86] is leveraging the leaf-spine network topology, while in terms of NFVO capabilities, only [52, 85, 89] provide LCM functionalities. Although multiple heuristics are provided for different optimization problems, all authors except [52, 89] consider a single administrative domain with a centralized datacenter, while authors in [52, 89] follow a core-edge architectural approach. Finally, [52, 83–87] provide heuristic-enabled placement algorithms, while [20] provides approximation placement algorithms.

5.2.2 Multi-domain VNF placement

Taking a deeper look into the SoA works for multi-domain environments, authors in [21] provide a decentralized multi-domain architecture. They design a cost-aware heuristic-guided algorithm as the embedding algorithm, while they evaluate their results using a custom made platform. While this work is the most complete recent multi-domain approach we have found in the literature to our knowledge, the proposed architecture does not provide any scaling/migration LCM functionalities. Additionally, their experimental results focus only on the scalability of their orchestration system. Authors in [22] provide a 5G OS in order to lower the complexity of the underlying 5G infrastructure in a multi-domain environment. Their work is limited, though, to a high-level architecture of the 5G OS, and the placement, orchestration or LCM techniques are not explained in depth. Finally, authors in [23] provide a multi-cloud orchestration solution that is completely decentralized. They provide three different optimal-based solutions for the VNF placement problem, focusing on cost, quality of experience (QoE) and the game theory based trade-off between the cost and QoE, respectively. This work does not support migration functionalities, while the simulation-based results are focusing on the performance evaluation of the proposed solutions in terms of the number of flavors and the number of datacenters' locations.

5.3 Multi-domain 5G architecture

In this section we provide the building blocks of the proposed multi-domain 5G architecture. Additionally, we provide the 3GPP and NFV compliance related information, the system model and the cost interpretation.

5.3.1 Cross-operator federation

The proposed architecture is composed of a federated network of local and foreign compute infrastructures, distributed across several points of presence (PoPs). For each domain, either the local or the foreign, it can be considered that the compute substrate is formed by a large centralized cloud datacenter complemented by a number of smaller decentralized MEC datacenters. The large cloud facility will be typically deployed at the core of the network, in well-connected central PoPs. On the other hand, the MEC hosts will provide more compute capabilities at the edge of the network.

While centralized cloud hosts can provide higher bandwidth and larger volumes of CPU, RAM and storage than the MEC counterparts, they usually experience greater latency for reaching end users. We can refer to local cloud (LCloud) and local MEC (LMEC), respectively, for each of those environments when deployed at the local domain, and, equivalently, foreign cloud (FCloud) and foreign MEC (FMEC) when referring to the infrastructures in the foreign domains. Note that a foreign domain could correspond to capabilities offered either by private infrastructure and network service providers and MNOs, or by public providers (e.g., Google Cloud, Amazon Web Services, etc). Figure 5.1 illustrates the federated local and foreign compute domains.

The service provider in the local domain will then dispose both centralized and decentralized compute capabilities at both local and foreign domains. From the topological point of view, this could be assimilated as an overlay leaf and spine architecture, suitable for scatter gather workloads, where the local domain network infrastructure facilities play the role of spine nodes while the foreign domain network infrastructure facilities behave as leaf nodes, as represented in Fig. 5.1.

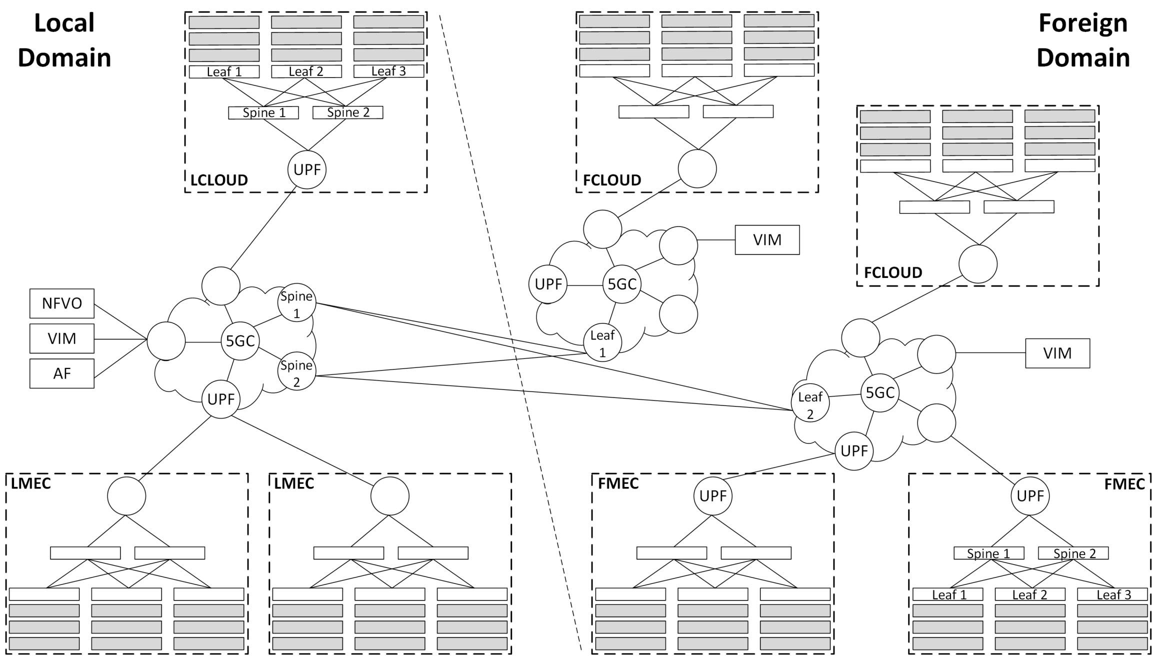

Figure 5.1: Reference topology for a federated Cloud-MEC architecture

The typical two-tier leaf and spine topology is full-mesh and consists of spine (i.e., L3 network layer) and leaf (i.e., L2 data link layer) switches. The spine layer is the backbone of the network and every leaf switch, that aggregates the traffic from the connected servers, is interconnected with every spine switch. Regardless the leaf that each server is connected to, the same number of devices will have to be crossed, every time it has to connect to another server (i.e., east-west) and the traffic needs to travel only through a spine and another leaf switch. This creates a low and predictable latency, while, in the case of failure of a spine switch, there would be only a slight degradation [93].

From the connectivity perspective, the local and the foreign domain network infrastructures will communicate with each other, since commonly they will be located at central parts with access to interconnection points between the local and foreign provider. In addition, both cloud and MEC datacenters, whether they belong to a local or foreign domain, they follow the leaf-spine architecture as well. In the case of MEC sites in either domain, they typically will require to go to the same interconnection points for communicating between local and foreign domains, thus the same idea of leaf and spine yet applies.

5.3.2 3GPP and NFV compliance

With respect to the 3GPP standard [28], we consider multiple 5GCs, where the 5GC NFs reside, and multiple UPFs across the overall infrastructure. One UPF can serve multiple cloud or MEC datacenters, while multiple UPFs can be used in order to direct the traffic towards any destination PoP. Additionally, we assume that the MEC solution aligns with the ETSI GR MEC 017 specification [94], allowing the integration with ETSI NFV MANO architecture in such a manner that ETSI NFV MANO can provide the necessary orchestration at distributed environments [95]. Additionally, the traffic steering capabilities, where the traffic is directed to the destination host, are provided by the AF NF [96].

Fully adopting the NFV paradigm, we consider that the 5G vertical applications are implemented in the form of successively interconnected VNFs that result in SFCs, as shown in Fig. 5.2. The VNFs that are deployed across the infrastructure have specific CPU, RAM and storage requirements, but also latency needs for the inter-VNF communication, in accordance with the application's QoS. Additionally, each SFC service is characterized by a min, max and mean lifetime duration [97]. The central controller of the system, the NFVO, is located in the local domain and is responsible to manage and orchestrate the federated resources, based on the 5G vertical applications' SLAs. In order for this to be achieved, local and foreign VIMs subscribe to the NFVO that issues the control commands towards the appropriate VIM, in order for the latter to perform the LCM actions. The NFVO is the module responsible for the placement of the building blocks of the SFCs across the federated network, with regard to the compute resources, but also to the latency SLAs that need to be met for the 5G vertical applications to run seamlessly. Finally, the NFVO controls the AF NF in order to apply the traffic steering rules.

According to the 5G paradigm [28], the UE, which is represented as user VNF in Fig. 5.2, receives services through a PDU session, which is a logical connection between the UE and the data network that provides the application. In order for the AF to submit the traffic steering rules to alter the PDU session, in the case of a local 5GC, the AF communicates directly with the PCF, as the AF is considered trusted. In the case of a foreign 5GC, the AF is considered as a foreign application and needs to interact with the NEF that securely exposes capabilities and events of the 5GC to third party applications, acting as the intermediate between the AF and PCF communication. Regardless of the source of the traffic steering request,

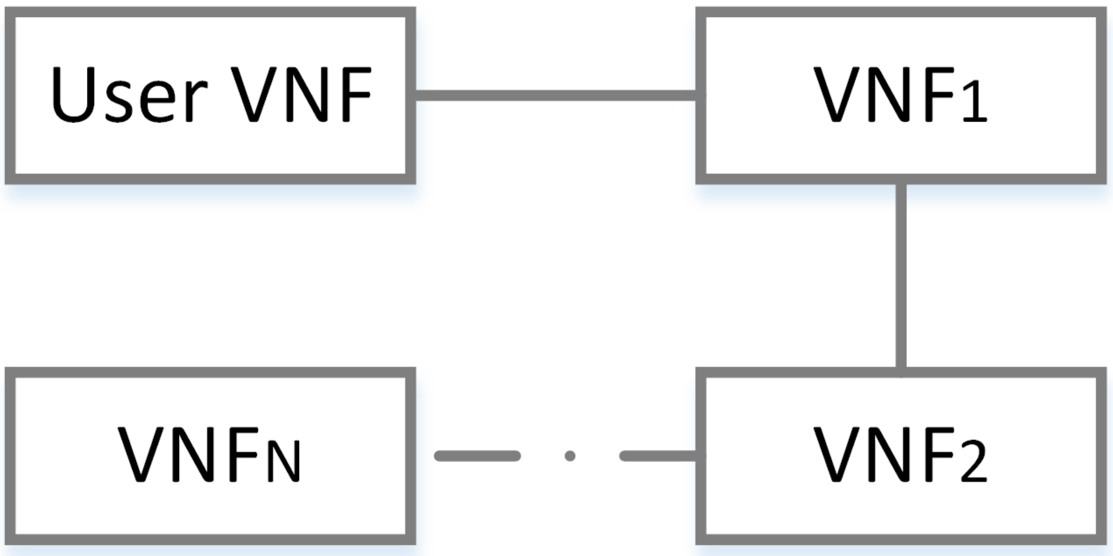

Figure 5.2: Service function chaining system model

the PCF submits the new traffic rules to the SMF that is responsible to translate it to uplink classifiers and pushes them to the chosen UPF, or UPFs when multiple UPFs mediate. Then, when the UPF detects the specific type of traffic from the UE, it steers the packets towards the next UPF that is included in the path until the final local break-out towards the destination data network of the PDU session.

5.3.3 System model

Figure 5.3 demonstrates the system model of the architecture. For each host, either in the cloud or the MEC datacenter, NFV support is provided. Additionally, a known capacity β is provided for each host h in terms of CPU, RAM and storage. Any two hosts can be connected through a maximum of 3 hops, where 0 hops represent the same node, 1 hop represents the same datacenter, 2 hops represent a different datacenter within the same domain and 3 hops represent any datacenter in any different domain. Moreover, the logical link between any two hosts h_s, h_d is defined by a characteristic latency b, with respect to the area of coverage. Finally, the leaf-spine topology provides the required stable latency between the links.

Each SFC consists of N VNFs, ($VNF_1, VNF_2, ..., VNF_N, N = |V|$), while VNF_0 is the user VNF. SFC request arrivals are characterized by a Poisson distribution. This probability distribution gives the probability of occurrence of events $(\tau; \lambda)$ in a fixed interval of time T provided that these events occur with a known constant

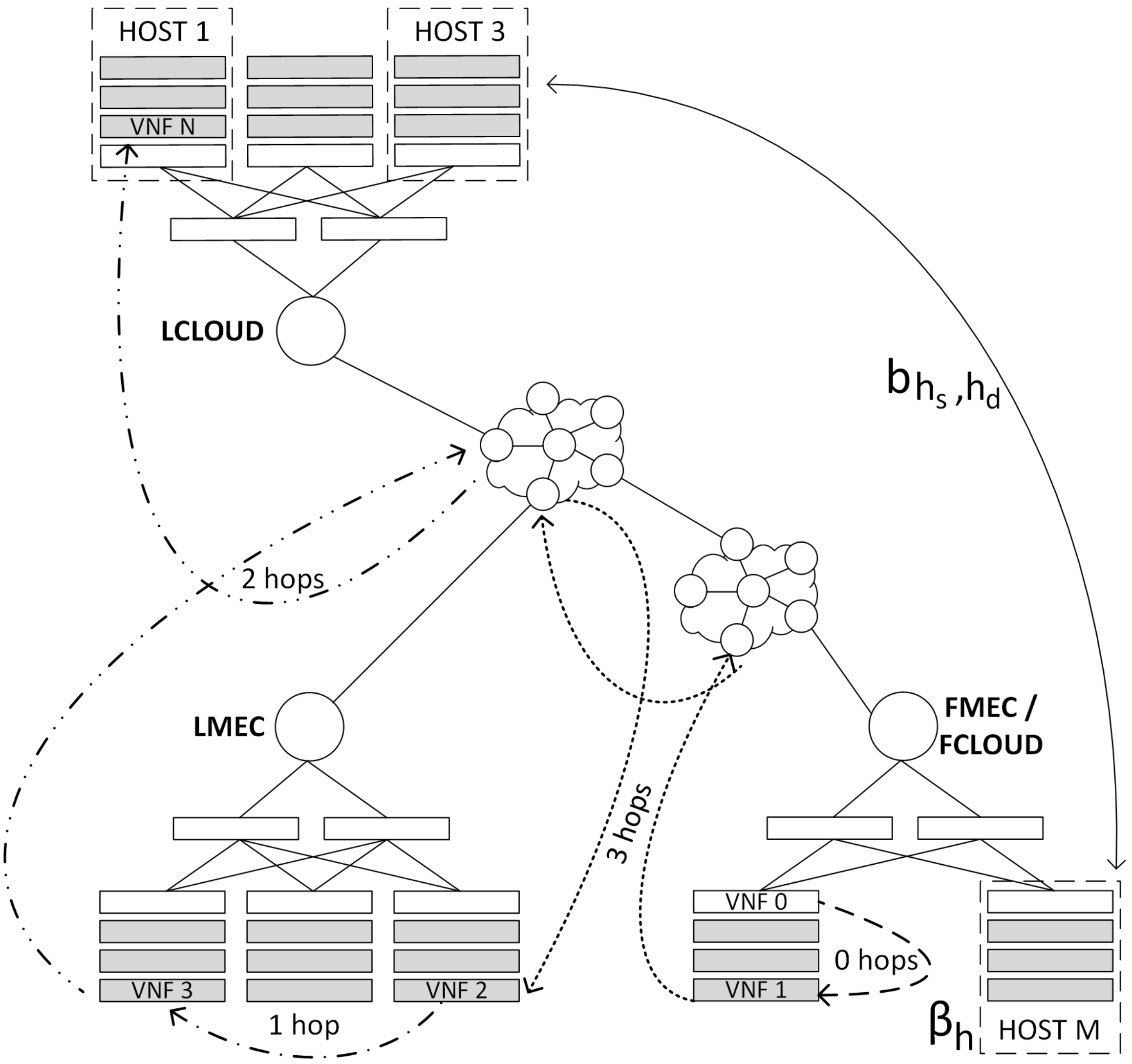

Figure 5.3: System model

rate λ and independent of the time t since the last event occurred, as indicated by (5.1).

$$P(t; \lambda) = \frac{e^{-\lambda}\lambda^t}{t!} \tag{5.1}$$

where:

- $\lambda \in \mathbb{R}+$ is the arrival rate of service requests based on the type of service, and
- $t \in \mathbb{R}+$ is the time of arrival.

Service lifetime is characterized by a truncated normal distribution to ensure that the duration belongs to a finite [min,max] time period. Equation (5.2) provides the service lifetime P, which is characterized by a mean μ and a standard deviation σ^2 that lies within the interval (a, b), where $-\infty < a \leq b < \infty$.

$$P(\chi; \mu, \sigma, a, b) = \begin{cases} \frac{f(\frac{\chi-\mu}{\sigma})}{\sigma(F(\frac{b-\mu}{\sigma})-F(\frac{\alpha-\mu}{\sigma}))}, \alpha \leq \chi \\ 0, \text{otherwise} \end{cases} \tag{5.2}$$

where:

- $\mu \in \mathbb{R}+$ is the mean service lifetime,
- $\sigma^2 \in \mathbb{R}+$ is the standard deviation which is set to 1 in order to ensure a small deviation from the mean,
- $f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}} z^2, z = \frac{x-\mu}{\sigma}$ is the probability density function of the normal distribution, and
- $F(z) = \frac{1}{2}(1 + erf(\frac{z}{\sqrt{2}})), z = \frac{x-\mu}{\sigma}$ is the probability density function of the cumulative normal distribution.

5.3.4 Cost interpretation

A cross-operator collaboration is beneficial both in CAPEX and OPEX costs. In the former, the MNOs can expand their coverage without the need of additional investments that would require network and compute equipment, just by renting on demand the already deployed infrastructure of a collaborative MNO. In more detail, the benefit of using a foreign provider is four-fold.

- First, the local provider can expand its area of coverage, without any CAPEX investment for equipment, simply by using on-demand the network/compute infrastructure of a collaborating foreign provider.
- Second, the local provider can use the foreign infrastructure (FMEC) in order to provide latency critical services to users that, due to their physical location, are not covered by an LMEC.
- Third, the local provider can use the foreign access network and the local infrastructure (LMEC/ LCloud) to provide latency tolerant services to the users.

- Forth, when the local resources are used at full capacity, new incoming services can be assigned to the foreign infrastructure, instead of being rejected. Those services can be migrated to the local infrastructure, when local resources are released.

In our work, we assume that the local infrastructure, owned by the local provider, is already deployed (i.e., zero CAPEX cost). We only focus on the OPEX cost, including the costs of maintenance, operation, energy consumption (e.g., for powering and cooling the servers) and hardware failure replacement. In our case, we consider the related OPEX cost to be included in the cost allocated per CPU, RAM and storage usage. In our current work, we emphasize on the compute resource investigation, while, in terms of network resources, our work focuses on the latency aspect, which is guaranteed by the leaf-spine topology of our architecture.

Furthermore, the service provider can make use of additional compute capabilities by leveraging on the foreign infrastructure. The advantage of this approach is that the local provider can expand its current infrastructure in a tailored way by using a metered service (also called pay-per-use or pay-as-you-go) where it has access to potentially unlimited resources but only pays for what it uses. Such a cost can be assimilated to be an OPEX cost. In a general way, the cost of each MEC resource will be more expensive than the ones at the centralized cloud, and the cost due to the usage of the foreign resources is higher than the local ones. Note that the foreign infrastructure use cost is also affected by a constant γ that is agreed between the local and foreign provider. From the local provider point of view, constant γ is the extra fee that needs to be paid to the foreign provider for using its resources, apart from the actual cost of using the CPU, RAM and storage of the host. From the foreign provider point of view, constant γ represents the actual profit of the foreign provider, when permission to the local provider to use its infrastructure is granted.

We define as cost of occupying EC_h^r at a host h, the cost of using r resources, where $r \in R, R = (CPU, RAM, storage)$ at the host for one minute. Furthermore, we define as VNF deployment CPM T_{hv} the amount of α_v^r resources, which are the resources that are demanded for that precise VNF, multiplied by their occupying cost per resource EC_h^r at the specific host h, adding the constant γ where applicable. Subsequently, the VNF execution cost is defined by the VNF deployment CPM multiplied by the actual time the VNF was running. Accordingly, we define as

SFC deployment CPM SFC_{Cost}, the cost of executing all the VNFs of an SFC at their respective hosts, and SFC execution cost the SFC deployment CPM multiplied by the actual time the SFC was executed.

In real scenarios, the actual SFC deployment and execution cost may greatly vary and depends on various factors, such as the energy efficiency of the underlying infrastructure (e.g., the energy consumption of the CPUs), the cooling methods, the location and the weather conditions of the datacenter. Taking into consideration the virtualization concept, the cost also can be affected by the virtual to physical CPU and RAM allocation ratio. This ratio defines the virtual CPU cores and RAM allocated per physical CPU core and RAM, respectively. A high ratio translates to more virtual CPU and RAM capacity per server, which leads to more VNFs running per server, resulting in a lower cost per service. Respectively, a lower ratio may result in more expensive services. Finally, the aforementioned cost is also affected by the actual traffic of the VNFs. A VNF with low traffic corresponds in low CPU utilization, and consumes, in general, less energy than a VNF with high traffic and CPU utilization. Therefore, depending on the expected traffic and resource utilization, the appropriate virtual to physical CPU and RAM ratio can be modified, affecting accordingly the per service deployment and execution cost.

5.4 SFC placement algorithms

In this section, we introduce two SFC placement algorithms. Our SFC placement algorithms are based on the cost minimization approach, with respect to the compute resource requirements of the VNFs and the latency SLAs for the inter-VNF communication. They are both online, as the placement decision is taken in order to satisfy end users' requests, under the resource and latency constraints, but also based on real-time data (e.g., actual server load and capacity). In what follows, we provide the analysis as an ILP formulation, whose solution generates the optimal cost-based placement of the SFC placement problem. Additionally, we propose a hop-based heuristic algorithm, as a less complex, thus faster, solution that approaches the accuracy of the optimal solution.

Table 5.1: Summary of key notations

V	Set of VNF instances requests, $v_1, v_2, ..., v_n, n = \lvert V \rvert$
C	Set of channels between VNFs
H	Set of hosts (PoPs), $h_1, h_2, \ldots, h_m, m = \lvert H \rvert$
L	Set of links between hosts, $l_{s,d} = (h_s, h_d) \in L$
S	Set of statically allocated user VNFs, $S \subset V$
R	Set of unique compute resources offered by hosts
R'	Set of unique network resources offered by links
α_v^r	Amount of resource r required by VNF v, $r \in R$
β_h^r	Amount of resource r available at host h, $r \in R$
c_{v_s,v_d}^r	Amount of resource r required for the v_s, v_d channel connection, $(v_s, v_d) \in V, r \in R'$
b_{h_s,h_d}^r	Resource r of the link between hosts $(h_s, h_d) \in H, r \in R'$
EC_h^r	Cost of occupying r resource at host h for $1', r \in R$
T_{hv}	Total cost of VNF v at host h for $1'$
DC_h	Set of local and foreign hosts

5.4.1 ILP formulation

With respect to the QoS, in terms of latency and compute resources, we propose a cost-aware ILP formulation, which is an NP-hard problem. We construct the SFC placement problem as a ILP optimization problem, in order to deliver an optimal solution for the minimization of the SFC deployment cost. In what follows, we provide the input of the ILP formulation, as well the constraints that are needed to define the minimum deployment cost, based on the actual host and link resources of the overall infrastructure.

Table 5.1 summarizes the key notations. Given the following parameters as input:

$R = CPU, RAM, Storage$

$R' = Latency$

DC_h

EC_r^h

S

The objective of the problem is to minimize the SFC deployment cost.

$Objective = minSFC_{Cost} = min \sum_{h \in H} \sum_{v \in V} A_{hv} T_{hv}$,
subject to $SFC_{Cost} > 0$

where:

$$DC_h = \begin{cases} 0, & \text{if } h \text{ is a local host} \\ 1, & \text{if } h \text{ is a foreign host} \end{cases} , \forall h \in H \tag{5.3}$$

$$A_{hv} = \begin{cases} 1, & \text{if VNF } v_k \text{ is deployed at host } h \\ 0, & \text{otherwise} \end{cases} , \forall h \in H, v \in V \tag{5.4}$$

$$\sum_{h \in H} A_{hv} = 1, \forall v \in V \tag{5.5}$$

$$\sum_{v \in V} A_{hv} \alpha_v^r \leq \beta_h^r, \forall r \in R, \forall h \in H \tag{5.6}$$

$$\sum_{v_s, v_d \in C} c_{v_s,v_d}^r \geq b_{h_s,h_d}^r, \forall h_s, h_d \in L, r \in R' \tag{5.7}$$

$$T_{hv} = \gamma DC_h + \sum_{r \in R} \alpha_v^r EC_r^h, \forall v \in V, \forall h \in H \tag{5.8}$$

- In (5.3), DC_h is a decision variable and equals to 0 if host h is a local host and to 1 if host h is a foreign host.
- In (5.4), A_{hv} is a decision variable and equals to 1 if the VNF$_\text{v}$ is assigned to host h, 0 otherwise.
- Equation (5.5) ensures that a VNF will be placed only in one host.
- Equation (5.6) ensures that the considered host has sufficient resources R to allocate to the VNF$_\text{v}$.
- Equation (5.7) ensures that the considered link (h_s, h_d) has enough resources R' required by the channel (v_s, v_d).
- In (5.8), T_{hv} is the VNF deployment cost. γ is a constant that defines the extra amount that has to be paid for the use of a foreign host.

Please note that the γ constant impacts the SFC deployment cost and affects the decision where each VNF will be deployed, as it is a cost minimization problem, but it does not affect the performance, as all possible combinations of hosts and VNFs will be calculated, regardless of the existence of the γ constant.

The set V represents the VNF instance requests of the SFC. In order to preserve the ordering of the VNFs, in an SFC with N VNFs (i.e., $VNF_1, VNF_2, \dots, VNF_N$) the $v_1 \in V$ corresponds to VNF_1 of the SFC, $v_2 \in V$ corresponds to VNF_2 of the SFC, … , $v_N \in V$ corresponds to VNF_N of the SFC. Additionally, each VNF has two connection points, except the user VNF and the last VNF of the SFC that have only one connection point. The connection points are the logical links that connect the successive VNFs, i.e., the user VNF with VNF_1, VNF_1 with VNF_2, … , and VNF_{N-1} with VNF_N, and are characterized by a latency value $c^r_{v_s,v_d}$. Thus, the ordering is also preserved via the connection points. In case the latency requirement between two successive VNFs $c^r_{v_s,v_d}$ is not met by the latency between the two underlying hosts $b^r_{h_s,h_d}$ (Eq. 5.7), the destination VNF v_d cannot be deployed at the specified host h_d.

ILP, in the general case, is NP-hard solvable [98]. Our problem is NP-hard solvable, since it is a variation of the (0-1) knapsack problem, which is NP-hard solvable as well [99].

Proof. Suppose a sample version of the ILP formulation with five (5) hosts $h_1, h_2, \dots, h_5, |H| = 5$, one (1) VNF $v_1, |V| = 1$ that requires two (2) CPUs $\alpha_1 = 2$ only and no latency constraints. Let's consider the Table 5.2 with the available CPUs per host, as well as the current cost per CPU if VNF v_1 were to be deployed at that host h. The VNF can only be deployed in one host. The problem can be formed as:

min $SFC_{Cost} = min(T_1A_1 + T_2A_2 + T_3A_3 + T_4A_4 + T_5A_5) > 0$,

where:

$A_1 + A_2 + A_3 + A_4 + A_5 = 1$,
$\beta_1A_1 + \beta_2A_2 + \beta_3A_3 + \beta_4A_4 + \beta_5A_5 + \geq 2$,
$A_h = 1$ if host 1 is selected ; $A_h = 0$, otherwise,

Table 5.2: Sample version of ILP formulation

Available CPUs	β_1	β_2	β_3	β_4	β_5
Cost per CPU	T_1	T_2	T_3	T_4	T_5

$h = 1, 2, 3, 4, 5.$

As it can be seen, the problem is mapped to a variation of the (0-1) knapsack problem, that calculates the minimum, instead of the maximum, cost, which is proven to be NP-hard solvable [99].

5.4.2 Hop-based heuristic SFC placement algorithm

The fact that finding the optimal solution is NP-hard, makes its use unsustainable as the system expands. Therefore, we are proposing a hop-based heuristic algorithm for the cost-aware SFC placement (Alg. 3).

The user VNF location is static and is given as input. First, the algorithm tries to plan the deployment of the next VNF (i.e., VNF$_1$) of the SFC at the same host (0 hops), taking into consideration the VNF's compute resource requirements, as well as the latency measured by the previous VNF (i.e., user VNF). In case there are sufficient resources, the resources are allocated for the deployment of the VNF. In case there are insufficient resources, the algorithm checks the availability of the resources of the hosts within 1-hop proximity. If more than one hosts that comply with the VNF's requirements are found, the host with the least cost for the deployment of the VNF is selected.

If no 1-hop hosts are found, the algorithm expands the search to the host within a 2-hop proximity. Similarly, if no 2-hop hosts are found the algorithm expands its search to the 3-hop proximity hosts. Please note that if all the hosts are visited and no host is found that can respect the VNF's resource requirements, the SFC request is rejected and the already allocated resources are freed. Otherwise (i.e., when the VNF is allocated to a host) the process then continues with the rest of the VNFs of the SFC, starting the search again from the 0 hops host (i.e., the host where the previous VNF of the SFC was deployed). The search is repeated until all the VNFs of the SFC are allocated to their respective hosts and the instantiation process begins.

Algorithm 3 Hop-based heuristic SFC placement algorithm.

```
Input:
$V$ number of VNFs of the SFC
$VNF[0]$, host where user VNF is placed
$VNF_{REQ}[0] = 0, 0$ requirements for the user VNF
$VNF_{REQ}[V]$ requirements in CPU/ RAM/ STORAGE per VNF of the SFC
$VNF_{SLA}[V_S, V_D]$ Latency requirements between two VNFs
$Host_{RES}[H]$, Available resources of CPU/ RAM/ STORAGE in every host
$Link_{SLA}[H_S, H_D]$, Available latency from host source to host
destination
$HOPS[H_S, H_D]$, Number of hops from host source to host destination
Output:
$SFC_{cost}$
$VNF[V]$ hosts where VNFs are placed
Initialization:
$SFC_{cost} = 0$
Execution:
for $v \in \{1, \dots, V\}$ do
  for $hop \in \{0, \dots, 3\}$ do
    $min_{cost}[vnf] = 10^{10}$
    $min_{host}[vnf] = -1$
    for $H_D \in \{1, \dots, HOSTS\}$ do
      if $HOPS[VNF[vnf-1], H_D] == hop$ then
        if $VNF_{REQ}[vnf] <= Host_{RES}[H_D]$ then
          if $VNF_{SLA}[vnf-1, vnf] \geq Link_{SLA}[VNF[vnf-1], hd]$ then
            Calculate $cost[hd, VNF[vnf]]$
            if $cost < min_{cost}[vnf]$ then
              $min_{cost}[vnf] = Cost$
              $min_{host}[vnf] = H_D$
            end if
          end if
        end if
      end if
    end for
    if $min_{host} \neq -1$ then
      $VNF[vnf] = min_{host}[vnf]$ !allocate resources
      $Host_{RES}[min_{host}[vnf]] -= VNF_{REQ}[vnf]$
      $SFC_{cost} += min_{cost}[vnf]$
      break;
    end if
  end for
  if $min_{host}[vnf] == -1$ then
    reject SFC request !no host found for the VNF
    free all already allocated resources
    exit;
  end if
end for =0
```

In terms of algorithmic complexity, the complexity analysis for the individual loops is:

- $O(n)$ for the first for loop,

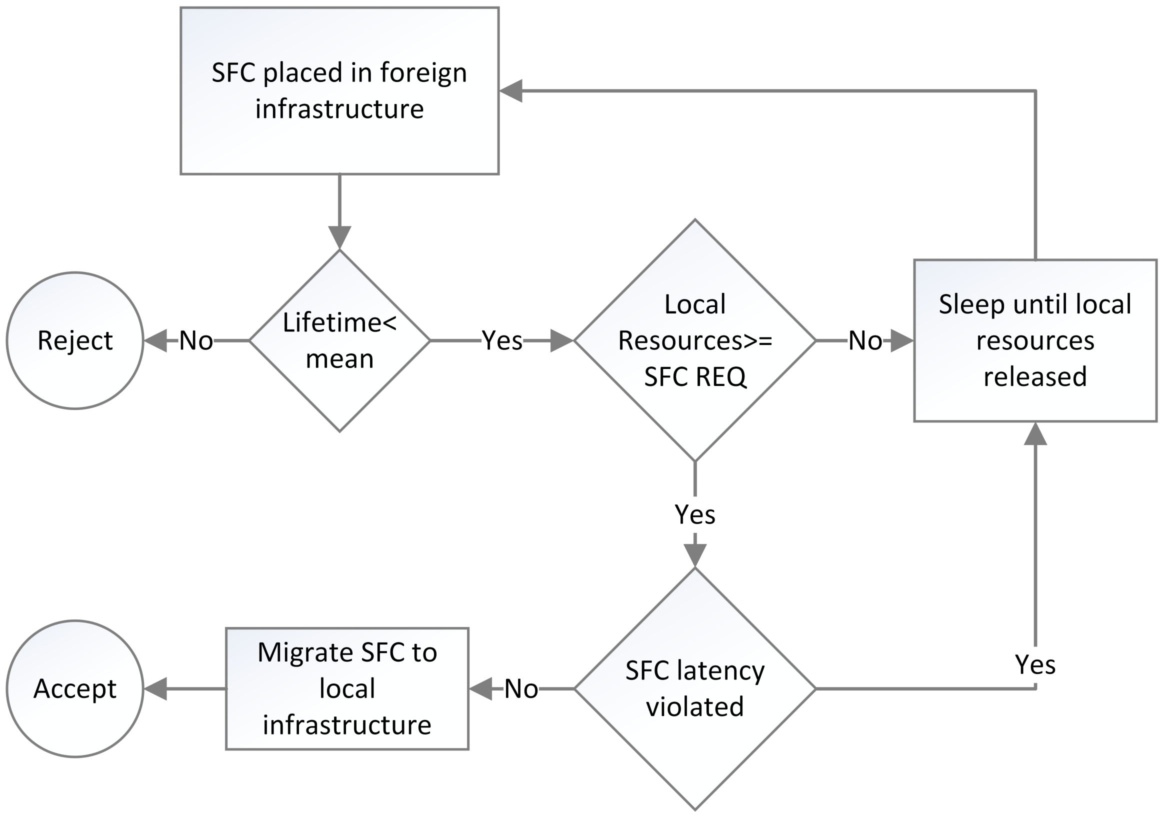
SFC placed in foreign infrastructure
Reject
No
Lifetime< mean
Yes
Local Resources>= SFC REQ
No
Sleep until local resources released
Yes
Accept
Migrate SFC to local infrastructure
No
SFC latency violated
Yes

Figure 5.4: Flow chart of the live migration algorithm

- $O(4)$ for the second (nested) for loop, and
- $O(n)$ for the third (nested) for loop.

This results in a $O(n)O(4)O(n) = O(n)1O(n) = O(n^2)$ overall complexity of the heuristic algorithm.

On the one hand, we conclude to the fact that the heuristic approach provides a less complex, thus faster solution, compared with the optimal one. On the other hand, one can conclude that the solution is only locally optimal, deploying, under some occasions, the VNFs in the foreign infrastructure when local infrastructure may also be a cheaper alternative. Hence, we provide a live migration operation, for the applicable cases, in order to further minimize the overall execution cost.

5.5 LCM functionalities

5.5.1 Live migration

The heuristic algorithm is faster than the optimal solution but its placement solution is optimal locally. In case the user VNF is located in the foreign infrastructure, the VNFs of the SFC will be most likely placed in the foreign infrastructure too, even if local infrastructure resources are available, which leads to an increased execution cost. The only exception that could lead to a globally optimal solution is when the SFC requirements are not met in the foreign infrastructure but are met in the local. For example, in the case of no available resources in the foreign infrastructure, the heuristic approach will place the SFC in the local infrastructure, in case the latency constraints are met. Thus, the solution will be the optimal.

In order to reduce the cost, we propose an online algorithm (Fig.5.4) that live migrates the SFC from the foreign to the local infrastructure. The migration step is initiated after the SFC is instantiated but could be executed either immediately, if resources and SFC latency requirements are met, or in a later time. In case of insufficient local resources or latency violations, the algorithm is put to sleep until a service that runs on the local infrastructure is finished and the resources it had allocated on the local host(s) are released. In order for the latter to be effective and reduce the cases where a service is migrated but shortly terminated, we only migrate services that their current lifetime has not exceeded their mean lifetime.

In terms of algorithmic complexity, a while loop, which checks that the actual lifetime of the SFC is smaller than the mean lifetime of the service, is implemented. The complexity analysis for the aforementioned while loop is $O(n)$, which results in a $O(n)$ overall complexity of the migration algorithm.

Let's consider the following C-V2X scenario, as a practical example where the benefit of live migration can be observed. We consider two services, e.g., the autonomous driving, which is considered latency critical, and the traffic jam warning, which is considered latency tolerant [100]. In case the vehicle (UE) is out of the premises of the local provider but within the premises of a foreign provider, the vehicle can use the access network of the foreign provider in order to request the aforementioned services and the user VNF will be considered as part of the nearest FMEC.

Based on the hop-based heuristic algorithm, the two SFCs needed for both services will be deployed on the FMEC (0 hops for same host or 1 hop for different host under the same MEC), if the resources are available. On the one hand, for the autonomous driving scenario this solution is the optimal as the latency constraint would have been violated if the SFC was deployed anywhere else. On the other hand, for the traffic jam warning scenario the solution is not optimal, as the SFC could have been deployed on the LCloud (or LMEC) without any latency SLA violation. Thus, the proposed migration algorithm will be activated, in the case of sufficient local resources, fixing the weakness of the locally optimal allocation of the latency tolerant service.

5.5.2 Scaling

The scaling functionality of our system is being triggered based on increased VNF load (e.g., due to increased traffic). In the case where any of the VNFs, which are part of the SFC, experiences high CPU load and the predefined scale-out threshold is exceeded, a new VNF is instantiated on the same host that the original VNF is located. This way, the latency requirement with the previous and the next VNFs of the SFC can be ensured. The traffic, and therefore the load, is equally shared between the two VNFs, using a load balancer (LB) VNF. Depending on the scenario, LB techniques such as round robin, least connection, resource based, etc., can be used. This way, we can resourcefully handle the increased load, in order to respect the QoS.

The scale-out process may occur as many times as the host resources allow it, increasing the VNF count of the SFC, but also the total cost of the SFC, accordingly. When the average CPU load decreases and reaches the scale-in threshold, the scale-in process occurs and terminates the last VNF that was created, as long as there are more than one. With this technique, we can efficiently regulate the system in order to release the excessively allocated resources when the load restores to normal, resulting in lower cost spending and better resource management.

5.6 5G experimental platform

In order to demonstrate a proof-of-concept of the described architecture, we have deployed an implementation of a 5G experimental platform (Fig.5.5). The hard-

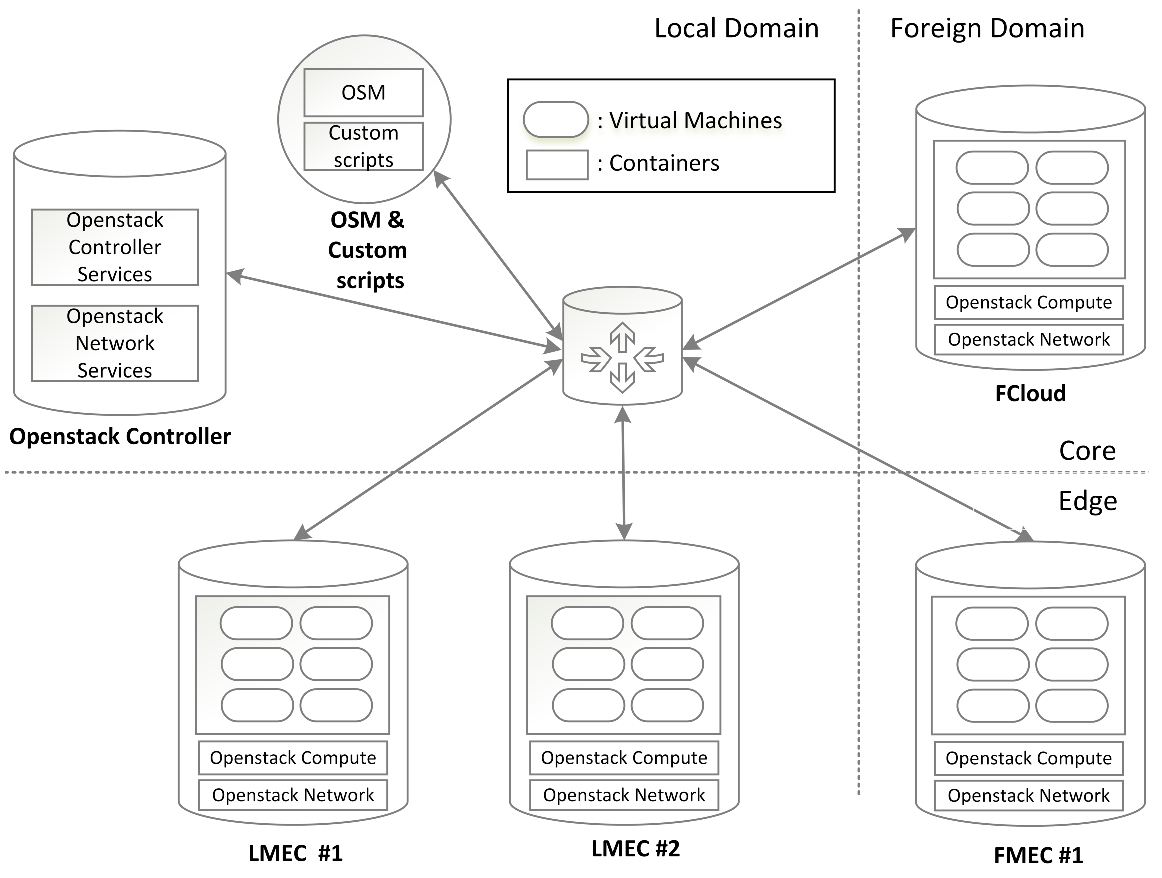

Figure 5.5: 5G experimental platform

ware of the platform (Table 5.3) consists of six general purpose computers allocated in a local and a foreign domain. Each domain is divided in a core and an edge location, where the cloud and MEC hypervisors are located, respectively. The cloud hypervisor has more compute resources, compared with the MEC hypervisors. In terms of network infrastructure, each machine has two 1 Gbps ETH network interfaces.

The chosen VIM of the experimental platform is openstack, which is an open-source IaaS provider. Openstack delivers the capabilities that enable the lifecycle actions (i.e., instantiation, scaling, migration, update/upgrade and termination) using VMs as the virtualization environment that VNFs are deployed. Openstack's multi-node deployment has been used with one controller node, where the controller and network services reside, and four compute nodes, both for the compute and network services of openstack, but also for the computing needs of the cloud and MEC environments. Opestack's services are deployed on LXD containers [75], achieving thus the required isolation from the VNFs that run as VMs at the com-

Table 5.3: Hardware characteristics

	Controller Node	Cloud Compute Node	MEC Compute Nodes x3	OSM
CPU	Intel i5-8500	Intel i5-8500	Intel i5-8400	Intel i5-7400
Cores	6	6	6	4
RAM	32GB	16GB	8GB	8GB
Storage	250 GB	250 GB	120 GB	120 GB
NIC	2x1Gbps ETH	2x1Gbps ETH	2x1Gbps ETH	2x1Gbps ETH

pute nodes. All nodes require two network interfaces, namely the management, for the communication of the openstack services, and the provider network, for the communication of the VNFs.

Openstack natively supports two features, namely the horizontal scaling, which is the expansion of the physical resources by adding more compute nodes, and the live migration of the VMs. The migration is defined as live, since it takes place without service interruption and the VM resumes from the exact same state as it was before the migration. Additionally, the duration of the migration might last from few seconds to few minutes, depending on the flavor of the VM instance (i.e., CPU, RAM and storage), the type of storage (i.e., ephemeral, object, block or shared file-based), the hypervisor type (e.g., KVM, QEMU, etc), the current load of the VM and many other variables. Nevertheless, there is no service disruption, as the old VM is running for the whole duration of the migration process and the traffic is switched to the new VM only after the latter is instantiated and ready to handle the requests.

The selected NFVO is OSM and is responsible for the orchestration of the underlying compute and network resources. OSM can achieve this by sending controlling commands to openstack, using the openstack APIs. Despite the fact that there are several available NFVOs [32], OSM is open-source, has a low resource footprint and provides a variety of operations that are sufficient for our demonstration purpose. Thus, we have selected it as the NFVO of the system.

OSM is configured via yaml descriptor files. The VNFD defines the needed VNF resources, in terms of CPU, RAM and storage, the VNF logical network connection points, the image of the VM, etc. The NSD defines the virtual links for the VNF interconnection, and maps them to the physical networks that are provided by the VIM. Additionally, OSM natively supports horizontal scaling, via VNFD

Table 5.4: SFC service characteristics

	Arrival rate	**Min lifetime (hrs)**	**Max lifetime (hrs)**	**Mean lifetime (hrs)**	**Number of VNFs**	**CPU cores**	**RAM (GB)**	**Storage (GB)**	**Revenue**	**Latency tolerance**
Service 1	12	0.5	6	3.5	2	2	2	4	r/4	High
Service 2	6	3	7	5	4	4	4	8	r/3	Medium
Service 3	4	4	9	6.5	6	6	6	12	r	Low

configurations, where the thresholds of i) the scale-out, ii) the scale-in, and iii) the minimum, and iv) the maximum number of VNFs, are set.

Both openstack and OSM are unaware of the 5G application that is running in the VMs. Additionally, OSM has no immediate control as to the compute node where the VM is executed, making it impossible both to guarantee the required QoS and respect the SLAs of the 5G application. This makes necessary the need for a middleware that enhances the intelligence of placement location. To that end, and in order to be able to select the compute node where the VMs are instantiated, we have applied our SFC placement intelligence, as custom-made scripts, therefore enhancing the capabilities of the orchestrator. The scripts are executed upon any incoming SFC request, and indicate the cost-aware placement of the VMs, based on Alg. 3.

5.7 Experimental analysis

In order to demonstrate the potential of the described architecture, we implemented an experimental setup leveraging the 5G experimental platform as described in section 5.5, and the CPLEX simulator. In this section, we provide the experimental setup and the various experiments we run.

5.7.1 Experimental setup

For our experiments, each VNF consists of a single flavor VM of 1 CPU, 1GB of RAM and 2GB of storage. Nevertheless, our 5G experimental platform supports custom flavors. Table 5.4 describes the characteristics of the three different SFC services in terms of arrival rate, lifetime, number of VNFs, VNF characteristics, relative revenue and latency tolerance [97]. Service 1 has the highest latency tolerance and arrival rate, but consumes the least resources and generates the lowest revenue. It could represent updating or navigation services, where latency is not strict. Service 2 has medium latency tolerance and arrival rate, medium resource consumption and medium revenue generation. This service could represent infotainment or non-emergency e-health services, where medium resources and moderate latency is needed. Finally, service 3 is the most latency strict, as it can tolerate low latency, and it has the lowest arrival rate, but it consumes the highest resources and generates the most revenue, compared with the other two services. Service 3 examples include automotive or industry 4.0 use cases, where the needed resources are high, while the latency that those services can suffer is low.

Beforehand, we created a sequence of SFC arrivals with their corresponding service times. The same sequence was used in all of our experiments. Please also note that the CPLEX simulator was used in order to find the optimal solution for the SFC placement, while the rest of the experiments run on our 5G experimental platform. The total duration of the analysis is four weeks (28 days), while in the same time period we observed 332 requests of service 1, 164 requests of service 2 and 111 requests of service 3.

In terms of deployment location, being able to suffer high latency, service 1 VNFs can be deployed in any datacenter in any foreign domain, which interprets as up to 3 hops, regardless the location of the user VNF. Additionally, service 2 VNFs can endure medium latency, so their deployment can take place up to a different datacenter within the same domain, to nodes that are up to 2 hops from the user VNF. Finally, service 3 VNFs can tolerate low latency only, so they can be placed up to the same datacenter, allowing them up to 1 hop from the user VNF. Both optimal and heuristic placement algorithms follow the cost optimization placement, as explained in section 5.5.

5.7.2 Iterations experiment

In the first experiment, we compare the iterations of the optimal (Fig. 5.6) and the heuristic (Fig. 5.7) solutions. An iteration is a loop, which is a sequence of commands that is repeated until a certain condition is reached. On the one hand, in the optimal solution, each one of the Eq. (5.4), (5.5), (5.6), (5.7) and (5.8), as well as the Objective, create loops that create equivalent iterations. On the other hand, in the hop-based heuristic algorithm, the iteration refers to the checks if a host is suitable for hosting the VNF. This sequence refers to the nested for loop in

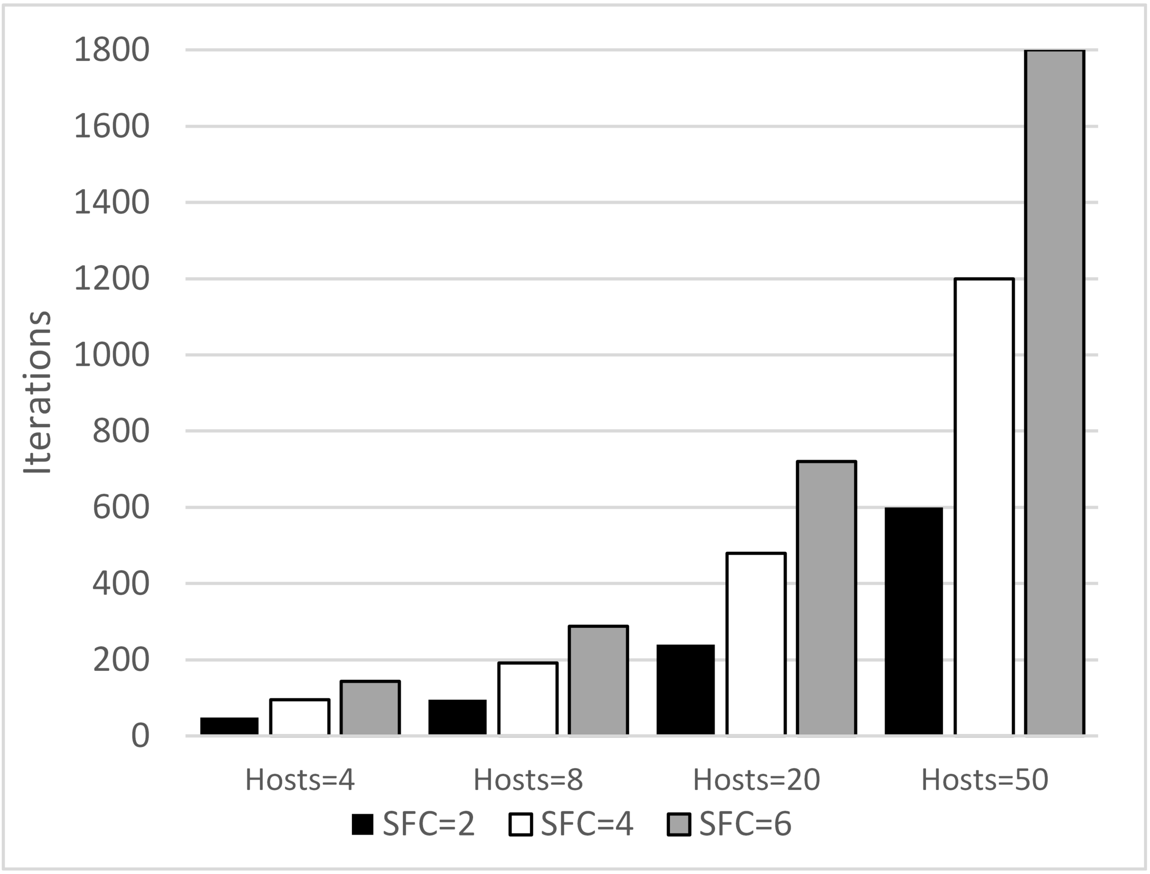

Figure 5.6: Average number of iterations for the optimal solution

line 5 of Alg. 3, and the sequence of commands in this loop (i.e., lines 6-16). A significant difference is that the optimal solution calculates the deployment cost of a VNF at all hosts and then selects the minimum cost based on the constraints. On the contrary, the heuristic algorithm calculates the deploying cost of the VNF only at the hosts that meet the compute resource and latency criteria, in a hop-based method, making it more efficient. We used the CPLEX simulator for the optimal solution and a custom python program for the heuristic.

Figures 5.6 and 5.7 display the optimal and the heuristic iterations, respectively, for two, four and six VNFs in the SFC and four, eight, twenty and fifty hosts in the system. The number of the iterations of the optimal solution is always the same, as all constraints need to be checked in order for the optimal solution to be found. On the other hand, our heuristic algorithm has a better overall response, with 66% less average iterations (e.g., 1800 average iterations for the optimal solution versus 600 average iterations for the heuristic solution). The maximum number of iterations

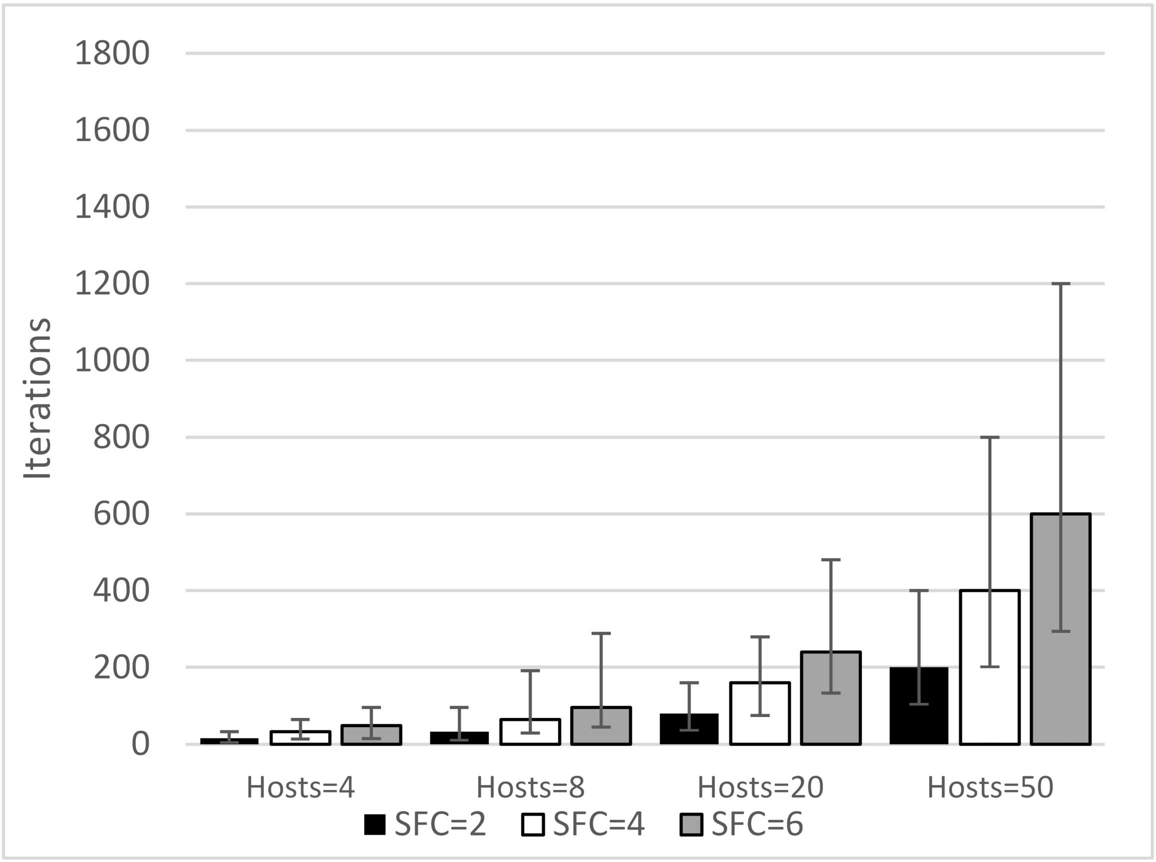

Figure 5.7: Average number of iterations the heuristic solution

(i.e., worst case scenario where all possible loops are executed) is still 33% less than the maximum number of iterations of the optimal solution (e.g., 1800 maximum iterations for the optimal solution versus 1200 maximum iterations for the heuristic solution). Additionally, the minimum number of heuristic iterations is equal to the number of the VNFs of the SFC, multiplied by the number of hosts, when they are deployed on the same host as the user VNF (0 hops). Finally, Fig. 5.6 demonstrates that the average number of iterations is increased faster for the optimal solution, since the optimal solution is a NP-hard problem, and slower for the heuristic (Fig. 5.7), thus making the latter more efficient in large scale environments.

5.7.3 Service block rate experiment

Given the predefined sequence of arrivals with their corresponding service times, we deployed and executed the SFCs on the 5G experimental platform, using the optimal placement solution provided by the CPLEX. Then, we run again the same

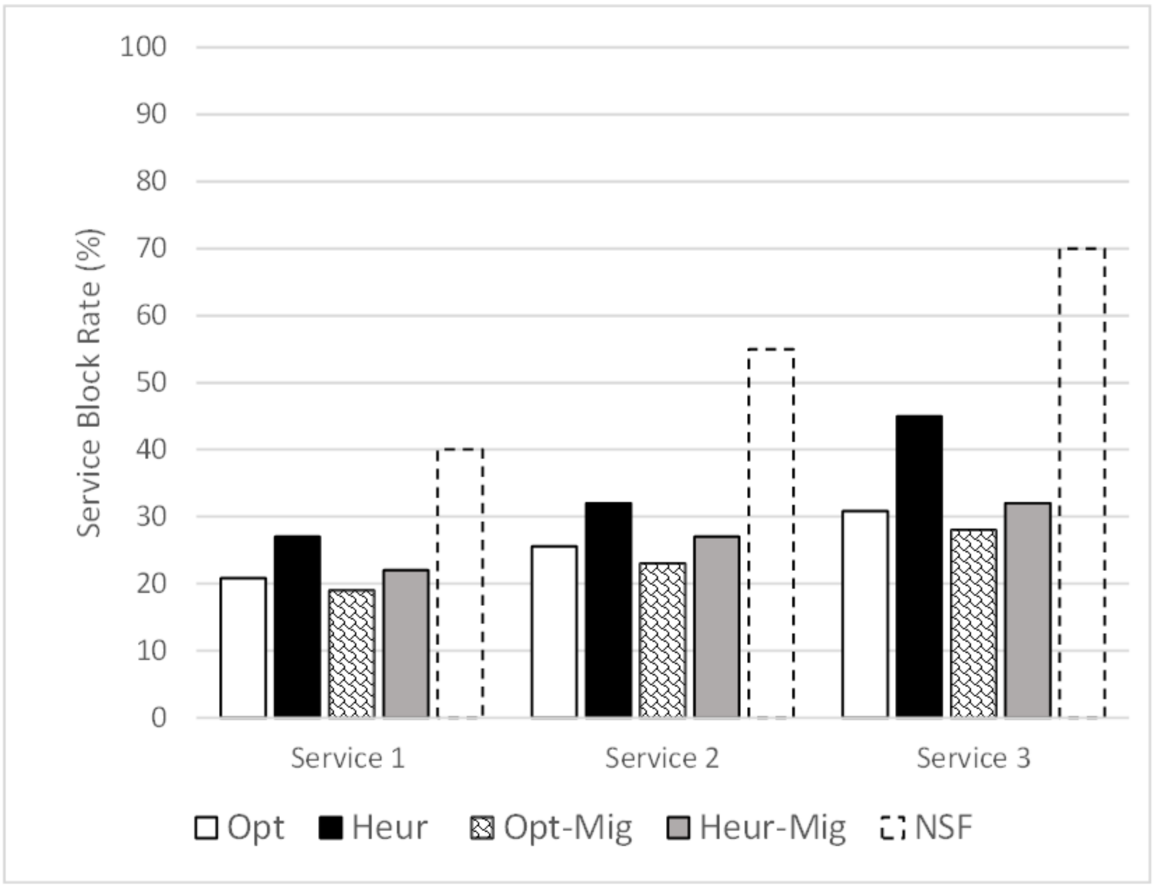

Figure 5.8: Block rate per service for the optimal (Opt) and the heuristic (Heur) placement method without the migration feature enabled, with the migration feature enabled (Opt-Mig and Heur-Mig), and comparison with the SoA (NSF)

sequence of arrivals, but this time we used the heuristic algorithm as the SFC placement method.

In Fig. 5.8 we can find the percentage of the block rate per service, for the two different placement methods (Fig. 5.8-Opt and Fig. 5.8-Heur), as well as the comparison with the SoA placement algorithm (Fig. 5.8-NSF). We use [83] to compare with the SoA, as their NSF heuristic algorithm uses similar logic with ours. For services 1 and 2, which have high and medium latency tolerance, respectively, we notice that the optimal solution has a slightly decreased service block rate, compared with the heuristic approach. For service 3, though, we notice that the block rate of the heuristic placement method is significantly higher, compared with the optimal placement. This is expected because the heuristic placement deploys the service 1 VNFs near their user VNF, regardless if the latter is located in the local or foreign infrastructure. The optimal placement approach (Fig. 5.8-Opt) deploys service 1 VNFs in the local infrastructure, if resources allow it, even in the case where the user VNF is located in the foreign infrastructure.

Furthermore, we demonstrate the block rate when the migration feature is enabled. While the gain for the optimal placement algorithm (Fig. 5.8-Opt-Mig) is low, the benefit can be observed for the heuristic placement method (Fig. 5.8-Heur-Mig), for example for the service 3, which is the service with the strictest latency requirements. By migrating the services with high latency tolerance (i.e., service 1) from the foreign infrastructure to the local one, apart from the reduced execution cost, we also save the foreign host resources. Therefore, there is a higher probability that a future service 3 request will be accepted by the foreign infrastructure, which is important because service 3 are the services that generate the most relative revenue, as depicted in Table 5.4.

In order for the migration to be effective and reduce the migration overhead, we only migrate VNFs whose current execution time has not exceeded their mean lifetime, as per Table 5.4. Finally, our heuristic solution with the migration feature enabled (Fig. 5.8-Heur-Mig) has a 45%, 49% and 54% reduced block rate for service 1, service 2 and service 3, respectively, compared with the NSF heuristic algorithm (Fig. 5.8-NSF) of the SoA. The reason why the NSF suffers a high block rate is because it is limited only to the resources of the local domain, without taking into consideration the resources of the foreign domain. Since there is no cross-domain collaboration, all incoming requests from a user VNF that is located in the foreign domain, are automatically blocked. The same applies to the incoming requests from a user VNF that is located in the local domain, when the local resources are fully occupied.

Live migration's primary benefit is to tangle with the locality of the heuristic approach. Nevertheless, it is also beneficial for the optimal approach, under the circumstances that all local resources are fully occupied. The decision of the SFC placement from the optimal solution is taken only once, without taking into consideration that some VNFs at local hosts might be terminated later and local, thus cheaper, resources are released and could be re-utilized by current VNFs at foreign hosts. The optimal solution itself is static, as the SFC deployment decisions are taken based on the current snapshot of the resources, but once the SFC deployment decision is taken, the optimal algorithm is terminated. With the migration function enabled, though, we manage to provide a dynamic solution that

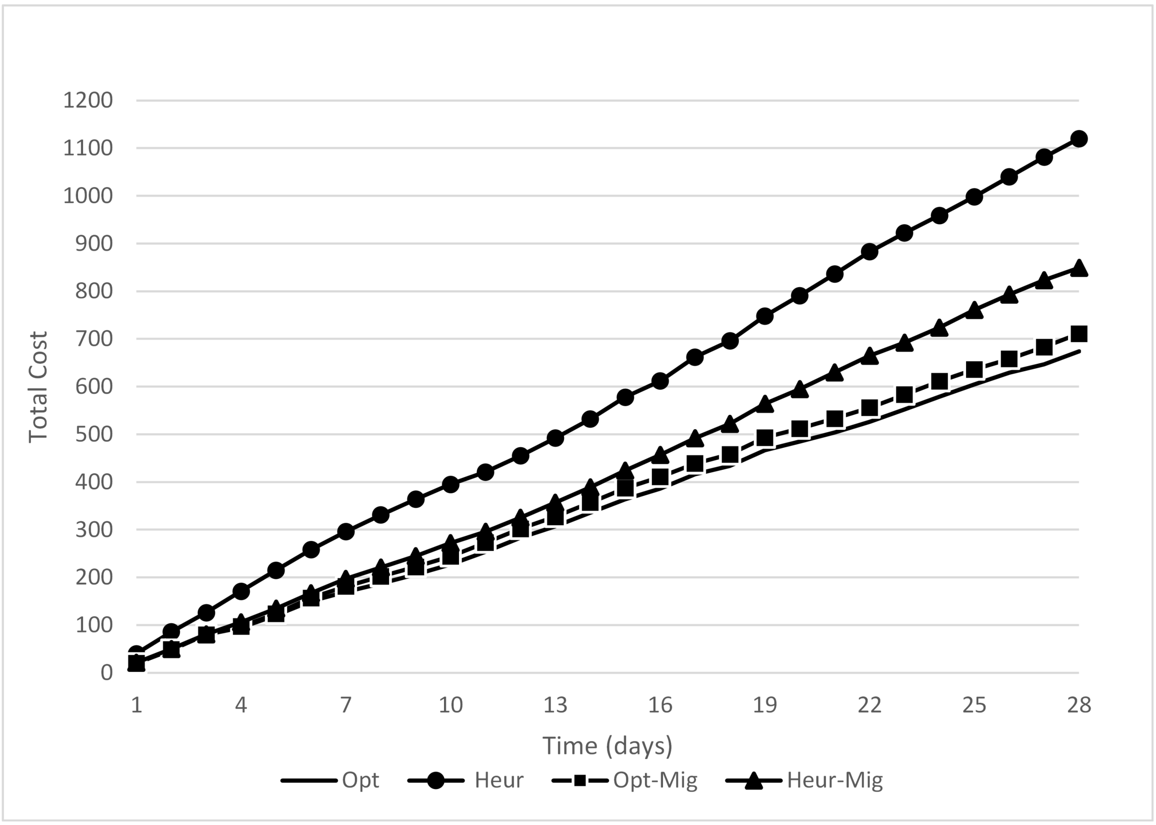

Figure 5.9: SFC execution cost for the optimal (Opt) and the heuristic (Heur) placement methods without the migration feature enabled, and with the migration feature enabled (Opt-Mig and Heur-Mig)

migrates the aforementioned VNFs in a later time, resulting in an improved, post-deployment cost-based resource reallocation.

5.7.4 SFC execution cost experiment

Under the same setup, we evaluate an additional metric, which is the total cost executing the SFCs using the optimal (Fig. 5.9-opt) and the heuristic (Fig. 5.9–heur) placement solutions respectively, in a 28-day period. As expected, the overall cost of the optimal is lower than the cost of the heuristic placement approach. Additionally, we run the same experiments but with the migration feature enabled. The results demonstrate a considerable gain on the heuristic placement method (Fig. 5.9-Heur-Mig), as the original heuristic was placing the VNFs based on the local best cost and not the global. With the migration feature, we migrate the VNFs from the foreign infrastructure to the local, which, by default, is more cost effective.

On the other hand, more service 3 requests are accepted, as explained earlier,

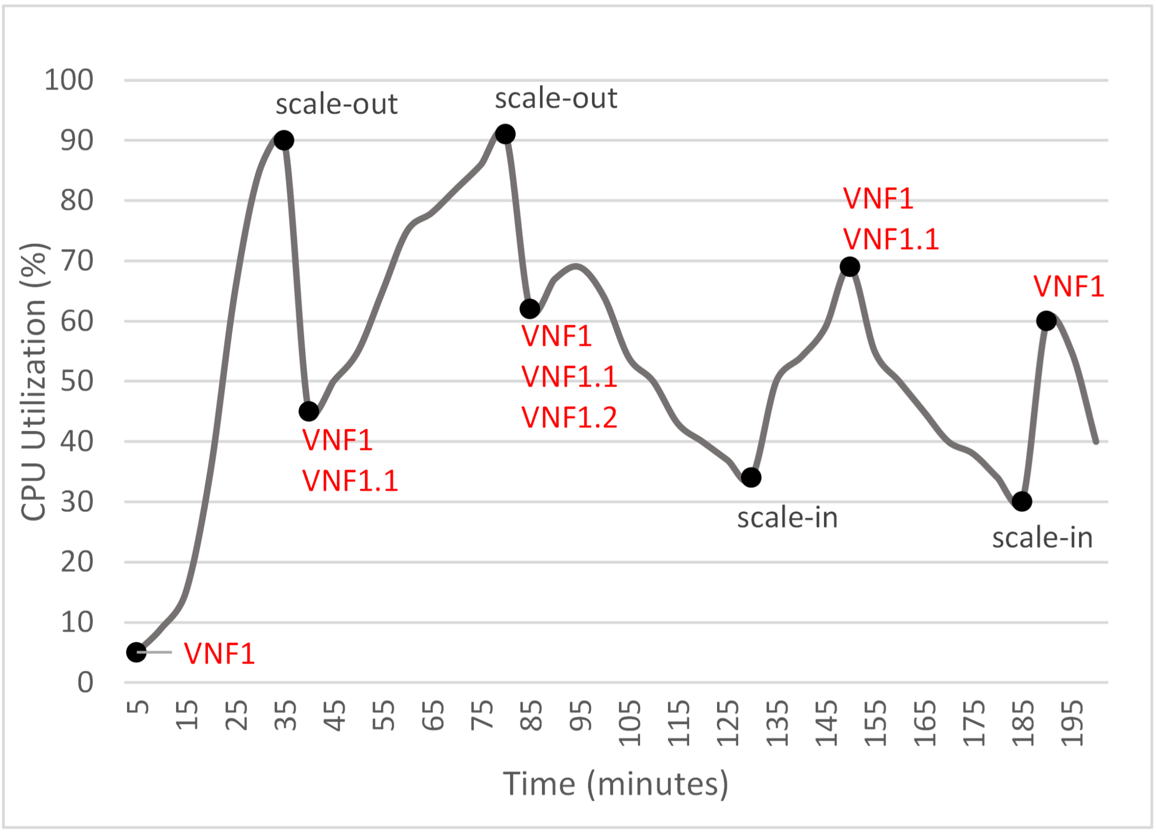

Figure 5.10: Scaling experiment

resulting in more VNFs to be placed in the foreign infrastructure. Finally, we notice a small difference also in the optimal placement method (Fig. 5.9-Opt-Mig). In this case, the migrations will occur for VNFs of service 1 that were originally placed in the foreign infrastructure, due to insufficient resources of the local infrastructure. This increase occurs because the service block rate is also decreased.

5.7.5 Scaling experiment

The scaling experiment demonstrates how a VNF of an SFC can scale-out, in case of increased CPU load, and scale-in upon reduced CPU load. In our experiment, we have set the scale-out threshold at 90% CPU utilization, the scale-in is set at 35%, while the round robin LB policy is selected. As shown in Fig. 5.10, when the scale-out threshold is reached, a new VNF is instantiated and the traffic is shared among the two VNFs. The same process occurs again, as the average CPU utilization of the two VNFs exceeds again the 90%, so a third VNF is instantiated and the traffic now is equally shared among the 3 VNFs. The reverse process (i.e., scale-in) takes place when the average CPU load reaches the 35% threshold, resulting in terminating the VNF that was instantiated last. This way, we can handle any sudden traffic peak without interrupting the service, in a resource efficient manner.

5.8 Conclusion

In this chapter we presented a novel multi-domain 5G architecture, able to exploit the interplay between the core and edge tiers in a federated environment. We discussed the key enabling technologies and filled the gap between the NFVO, the VIM and the LCM functionalities by proposing cost-aware SFC placement algorithms, with enhanced SFC LCM actions. We applied our algorithms to a fully deployed 5G experimental platform, where applications with different QoS and SLA constraints have been executed. The conducted experiments show that through the proposed heuristic placement method, a near-optimal cost-effective solution can be found, with 66% less average iterations that leads to decreased complexity, compared with the optimal placement method. Additionally, our experiments demonstrated an up to 54% reduced service block rate, compared with the SoA solution.

Chapter 6

Conclusions and Future Challenges

6.1 Conclusions

One of the greatest challenges for the 5G network operation and expansion in the near future is the accurate and efficient SFC placement and management, across environments that may present diversity or, even, multiple ownership. Since the vertical applications demand increasing resources at every level, up to the user proximity, it is imperative for the underlying 5G network infrastructure to be equipped with the intelligence required to expand and adapt to the traffic variations and unforeseen resource needs. However, several questions arise. For instance:

- What kind of architectures can be employed to tackle general or specific 5G use cases?
- How the VNFs can be employed in order to provide the required resources and isolation that the 5G use cases need?
- Where the VNFs will be placed and how can this affect the performance of the applications they host?
- How can the VNFs be managed in an efficient way throughout their lifecycle to cover the constantly changing needs of the UEs?
- In what way can MNO collaborations be beneficial both for users and operators?
- How can the CAPEX and OPEX costs of the MNOs be reduced?

In this thesis, we have made an attempt to answer these questions. We have described a variety of 5G architectures, able to benefit from the interplay between the core and the edge resources. The reason why we chose the specific architectures is because we wanted to support the demanding 5G use cases. In more detail, the fog-enabled architecture can support the automotive vertical applications, the IoT-enabled architecture can support IoT use cases, and the federated architecture can assist towards the execution of the demanding applications that will arrive from combing different vertical industry applications. Combined with our proposed VNF scheduling and placement algorithms, and with the aid of LCM techniques, the 5G use cases could be executed efficiently, adapting to realistic circumstances. The 5G experimental platform we deployed demonstrated the effectiveness of such techniques.

In the first part of the thesis, in Chapter 3, we presented a fog-enabled architecture with three layers of resources, able to deal with the resource intensive C-V2X applications of the automotive vertical. In addition, we provided a novel representation of how applications can be developed as SFCs, where the combination of VMs, containers and unikernes creates AaaSFCs. The LCM intelligence of migration that was introduced, facilitated handover scenarios, without any service disruption, even in the case that the vehicle was leaving the original service area, as mobility was taken into consideration. Additionally, the LCM action of scaling that was employed, enabled the service to adapt to high traffic patterns without service degradation.

Following, in the second part of the thesis, in Chapter 4, we shifted our focus to the online scheduling and LCM of chained VNFs. We demonstrated with a variety of experiments that our innovative online VNF scaling and live-migration scheduling algorithm can have a beneficial impact on the on-the-fly optimization of MEC and cloud resources. The application of our methods in our experimental platform lead to an increased number of latency critical IoT applications handled by the system, without SLA violations.

Finally, in the third part of the thesis, in Chapter 5, we introduced a federated environment, where MNOs collaborate and share their underlying infrastructure, in a way that is beneficial for all three stakeholders; the local provider, the foreign provider and the end user. We further extended the system by proposing cost-aware SFC placement and LCM algorithms that were applied on our 5G experimental platform. The extensive experimental analysis demonstrated significant improvements at the SFC placement efficiency, the service block rate, the SFC execution costs and the traffic peek handling.

Our thesis has contributed to the 5G wireless network enhancement by providing efficient mechanisms to support diverse 5G use cases in heterogeneous environments, where the VNFs are efficiently allocated between core and edge resources. We provided applied solutions in order to maximize the crucial UE requests that are accepted by the system, in a cost-efficient, for the MNOs, manner. Nevertheless, we take into consideration the SLA of each 5G use case and the various traffic and load patterns that UEs cause.

Concluding, this thesis has advanced the state of the art first by investigating the potential for benefiting from the optimal interplay between core and edge resources, able to increase the users served, respecting at the same time the 5G applications SLAs, and second by proposing SFC placement and LCM methods that are choosing the set of optimal locations for a chains of VNFs. Our solid work has been demonstrated through a series of experiments that benefits from our proposed algorithms that take advantage of the LCM techniques to manage and control the VNFs of the system, from their instantiation up to their termination. It is evident that there is still a lot of research ahead and our contribution is one of the sparks that will ignite new research towards the next generation wireless networks.

6.2 Future challenges

The evolution of the mobile networks is a non-stop process. Since the first generation (1G) that appeared around the 1980s, almost every ten years we have the next generation of the mobile networks [101]. The 1G mobile wireless communication network was analog used for voice calls only, while the second generation (2G) is digital and supports text and multimedia messaging. The 3G which provides multimedia support along with higher data transmission rates, while the 4G delivers much faster speeds, supports more intensive mobile activity, and reduces the cost of resources. The 5G promises to deliver higher multi-Gbps peak data speeds, ultra-low latency, massive network capacity, increased availability and better reliability, capabilities that the sixth generation (6G) technology will try to further improve.

Taking into consideration this continuous evolution, our work has contributed towards the improvement of the 5G technology. Despite the importance of our work, additional challenges need to be addressed towards the dominance of the 5G networks, but also towards investigating further the 6G technology.

6.2.1 Resource allocation

One issue that has risen is the network resource allocation across the system where SFCs are guaranteed to have sufficient resources to respect SLA. This is an issue that network slicing is trying to solve. There could be dedicated end-to-end slices, where fixed network resources are allocated to critical SFCs following a specific path, whether they constantly use the maximum amount of resources or not, or non-critical slices that share the network resources with multiple services using, for example, best-effort approaches, similar to the 4G concept [102]. Such network slicing techniques need to be selected, modified and studied in combination with our proposed solutions.

The physical to virtual resource allocation ratio, in terms of CPU, RAM and storage, is an area that needs to be further studied. This ratio can greatly affect the performance of the entire system. On the one hand, the physical resource underutilization could lead to increased CAPEX or OPEX costs, as unneeded dedicated resources will be allocated to VNFs. On the other hand, the overutilization of the resources could lead to service degradation or outage, as multiple VNFs will compete to access limited resources. In order to improve this topic, traffic prediction algorithms and dynamic physical to virtual resource allocation mechanisms could be further investigated.

6.2.2 Mobile network operator collaboration

An MNO collaboration includes technical tasks that need to be taken care of, such as additional connections (e.g., fiber) for connecting the datacenters of the different MNOs. In addition, 5G NFs can be provided by different vendors, such as Nokia, Ericsson and Huawei. Although the APIs that they use should be universal, this is not always the case. Thus, additional investigation may be required in order to provide a cross-operator platform, able to interconnect and manage the 5G NFs from the different vendors.

Furthermore, new business models and agreements need to be defined. Being a natural oligopoly, it is normal for the operators to be critic when it comes to collaborations with "rival" enterprises. Although synergies are not new (e.g., roaming services is the proof that operators can collaborate), MNOs need to discuss and build a common plan that would be satisfactory profitable for all stakeholders. Moreover, when it comes to CAPEX and OPEX cost savings, in an industry that can barely support the existing network demand, it is worth to investigate, business-wise, how such alliances could be implemented.

Finally, adopting a collaborative model could have legal implications. The federation of the underlying resources may extend out of the national borders and national legislation. This arises the issues of security and privacy, since the user data may need to travel and be stored in multiple countries with conflicting user data protection laws. Legal-wise, further investigation is needed in order to decide whether the current legislation of the countries that the involved MNOs operate allows such collaborations, or if changes need to be performed.

6.2.3 6G enabling technologies

The telecommunication technology never stops evolving. Driving autonomous systems, unmanned aerial vehicles (UAVs) and 16K resolution video-type data traffic will be just some of the 6G use cases. The driving force technologies for the 6G realization include the artificial intelligence (AI) and the application of machine learning (ML) algorithms [103], the 3D networking [104], and the optical wireless communication (OWC) [105].

ML is a subset of AI that creates algorithms and statistical models to perform a specific task without using explicit instructions, relying instead on patterns and inference. ML and AI allow 6G networks to be predictive and proactive. 6G speeds up the services on the cloud while AI analyzes and learns from the same data faster. The three main ML methods are the supervised, which relies on known models and labels that can support the estimation of unknown parameters, the unsupervised, where it relies on the input data itself in a heuristic way, and the reinforced learning, which relies on a dynamic iterative learning and decision-making process.

With the integration of ML and AI with MEC, MNOs can provide i) high automation levels from the distributed ML and AI architecture at the network edge, ii) traffic steering across access networks that is based on the application, and iii) dynamic E2E network slicing to address various scenarios with diverse QoS requirements. The primary focus of AI integration is to reduce CAPEX, to optimize the network performance, and to build new revenue streams. Additionally, novel computational intelligence models (e.g., neural networks, swarm Intelligence, etc.) and optimization algorithms (e.g., ant colony optimization, competitive imperialistic, etc.) can be employed to further improve the performance even of the 5G network system. Therefore, there is a new research opportunity on how AI/ML techniques can be applied to 5G networks.

Finally, 3D network models will enable ultra-fast dense networks where the terahertz (THz) frequency will be used in order to provide a fully digital, intelligent and connected world. 3D networks envision a 3D coverage where terrestrial platforms will be supported by non-terrestrial devices, like UAVs, drones or satellites. Additionally, OWC is a form of optical communication in which unguided visible, infrared or ultraviolet light is used to carry the signal, supporting thus the 3D coverage. Some interesting topics for investigation include the ways that the aforementioned technologies could be applied, their advantages and their limitations. As an intermediate step towards the 6G realization, the aforementioned technologies could be enablers for the beyond 5G networks (B5G).

Bibliography

[1] J. Peisa, "5G Techniques for Ultra Reliable Low Latency Communication," Ericsson, 2017. [Online]. Available: http://cscn2017.ieee-cscn.org/files/2017/08/Janne_Peisa_Ericsson_CSCN2017.pdf

[2] Ericsson White Paper, "5G wireless access: an overview," April 2020. [Online]. Available: https://www.ericsson.com/498a10/assets/local/reports-papers/white-papers/whitepaper-5g-wireless-access.pdf

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[4] ETSI, "Network functions virtualisation (NFV); management and orchestration," December 2014, Group Specification. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[5] A. J. Gonzalez, G. Nencioni, A. Kamisiński, B. E. Helvik, and P. E. Heegaard, "Dependability of the nfv orchestrator: State of the art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3307–3329, 2018.

[6] S. Sharma, R. Miller, and A. Francini, "A cloud-native approach to 5g network slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 120–127, 2017.

[7] ETSI NFV, "MEC in 5G networks," June 2018, White Paper No. 28. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[8] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *2014 Federated Conference on Computer Science and Information Systems*, 2014, pp. 1–8.

[9] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[10] OpenFog, "OpenFog reference architecture for fog computing," February 2017. [Online]. Available: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf

[11] 5GAA, "Report: Cost analysis of v2i deployment," August 2020. [Online]. Available: https://5gaa.org/wp-content/uploads/2020/09/5GAA_Ricardo-Study-V2I-Cost-Analysis_Final_110820.pdf

[12] EBU, "Technical review: Cost analysis of orchestrated 5g networks for broadcasting," March 2019. [Online]. Available: https://tech.ebu.ch/docs/techreview/EBU_Tech_Review_2019_Lombardo_Cost_analysis_of_orchestrated_5G_networks_for_broadcasting.pdf

[13] GSMA, "Operator Platform Concept," January 2020. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2020/02/GSMA_FutureNetworksProgramme_OperatorPlatformConcept_Whitepaper.pdf

[14] 5GPPP, "View on 5G Architecture," October 2021. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2021/11/Architecture-WP-V4.0-final.pdf

[15] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: Vnf placement algorithms for a dynamic amp; realistic edge cloud environment," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[16] D. B. Oljira, K. Grinnemo, J. Taheri, and A. Brunstrom, "A model for qos-aware vnf placement and provisioning," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 1–7.

[17] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2008–2025, Aug 2017.

[18] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of nfv service chains across geo-distributed datacenters," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 699–710, April 2018.

[19] H. Tang, D. Zhou, and D. Chen, "Dynamic network function instance scaling based on traffic forecasting and vnf placement in operator data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 530–543, March 2019.

[20] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 172–185, 2020.

[21] B. Sonkoly, R. Szabó, B. Németh, J. Czentye, D. Haja, M. Szalay, J. Dóka, B. P. Gerő, D. Jocha, and L. Toka, "5g applications from vision to reality: Multi-operator orchestration," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1401–1416, 2020.

[22] S. Dräxler, H. Karl, H. R. Kouchaksaraei, A. Machwe, C. Dent-Young, K. Katsalis, and K. Samdanis, "5g os: Control and orchestration of services on multi-domain heterogeneous 5g infrastructures," in *2018 European Conference on Networks and Communications (EuCNC)*, 2018, pp. 1–9.

[23] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal vnfs placement in cdn slicing over multi-cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 616–627, 2018.

[24] ETSI Group Specification, "Multi-access Edge Computing (MEC); Framework and Reference Architecture," January 2019, GS MEC 003 V2.1.1. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf

[25] B. Thekkedath, *Network Functions Virtualization for Dummies.* Hewlett Packard Enterprise Special Edition, Wiley, 2016.

[26] ETSI Group Specification, "Network Functions Virtualization (NFV); Architectural Framework," October 2013, GS NFV 002, V1.1.1. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf

[27] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1409–1434, Secondquarter 2019.

[28] ETSI Group Specification, "System architecture for the 5G system (5GS)," October 2020, Techical Specification. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

[29] Global mobile Suppliers Association, "5G Stand Alone Global Market Status: Executive Summary," January 2022. [Online]. Available: https://gsacom.com/paper/executive-summary-5g-standalone-january-2022/

[30] GSMA, "5G implementation guidelines: NSA option 3," February 2020. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2019/03/5G-Implementation-Guidelines-NSA-Option-3-v2.1.pdf

[31] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *2017 Global Internet of Things Summit (GIoTS)*, 2017, pp. 1–6.

[32] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2961–2991, Fourthquarter 2018.

[33] F. Bonomi, "Connected vehicles, the internet of things, and fog computing, Cisco, Eighth ACM International Workshop on VehiculAr Inter-NETworking (VANET)," 2011.

[34] J. Boleng, "Automated Provisioning of Cloud and Cloudlet Applications," 2013. [Online]. Available: https://resources.sei.cmu.edu/asset_files/Presentation/2013_017_001_48076.pdf

[35] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[36] P. L. Vo, M. N. H. Nguyen, T. A. Le, and N. H. Tran, "Slicing the edge: Resource allocation for ran network slicing," *IEEE Wireless Communications Letters*, vol. 7, no. 6, pp. 970–973, 2018.

[37] "The Openstack foundation." [Online]. Available: https://openstack.org/

[38] "VMware." [Online]. Available: https://www.vmware.com/

[39] "Kubernetes." [Online]. Available: https://kubernetes.io/

[40] "Open Source MANO." [Online]. Available: osm.etsi.org

[41] "ONAP." [Online]. Available: https://www.onap.org/

[42] "Minikube." [Online]. Available: https://minikube.sigs.k8s.io/docs/start/

[43] "Lightweight kubernetes." [Online]. Available: https://k3s.io/

[44] ETSI, "OSM IN ACTION," July 2021. [Online]. Available: https://osm.etsi.org/images/OSM_EUAG_White_Paper_OSM_in_Action.pdf

[45] Ericsson, "Ericsson mobility report," June 2019. [Online]. Available: https://www.ericsson.com/49d1d9/assets/local/reports-papers/mobility-report/documents/2019/ericsson-mobility-report-june-2019.pdf

[46] ETSI, "Digital cellular telecommunications system (phase 2+) (GSM); universal mobile telecommunications system (UMTS); LTE; 5G; Release 14," 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/121900_121999/121914/14.00.00_60/tr_121914v140000p.pdf

[47] ETSI, "5G; Vehicle-to-everything (V2X); Media handling and interaction (3GPP TR 26.985 version 16.0.0 Release 16)," November 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/126900_126999/126985/16.00.00_60/tr_126985v160000p.pdf

[48] NHTSA, "Traffic safety and the 5.9 GHz spectrum," June 2019. [Online]. Available: https://www.nhtsa.gov/speeches-presentations/traffic-safety-and-59-ghz-spectrum

[49] 5GAA, "C-V2X use cases: Methodology, examples and service level requirements," June 2019. [Online]. Available: https://5gaa.org/wp-content/uploads/2019/07/5GAA_191906_WP_CV2X_UCs_v1-3-1.pdf

[50] C. Bockelmann, N. K. Pratas, G. Wunder, S. Saur, M. Navarro, D. Gregoratti, G. Vivier, E. De Carvalho, Y. Ji, Stefanović, P. Popovski, Q. Wang, M. Schellmann, E. Kosmatos, P. Demestichas, M. Raceala-Motoc, P. Jung, S. Stanczak, and A. Dekorsy, "Towards massive connectivity support for scalable mmtc communications in 5g networks," *IEEE Access*, vol. 6, pp. 28 969–28 992, 2018.

[51] ETSI, "MEC deployments in 4G and evolution towards 5G," February 2018. [Online]. Available: https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp24_mec_deployment_in_4g_5g_final.pdf

[52] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online vnf lifecycle management in an mec-enabled 5g iot architecture," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4183–4194, 2020.

[53] H.-C. Hsieh, C.-S. Lee, and J.-L. Chen, "Mobile edge computing platform with container-based virtualization technology for iot applications," *Wireless Personal Communications*, vol. 102, no. 1, pp. 527–542, Sep 2018.

[54] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1380–1392, 2021.

[55] F. E. da Silva Barbosa, F. F. M. Júnior, and K. L. Dias, "A platform for cloudification of network and applications in the internet of vehicles,"

Transactions on Emerging Telecommunications Technologies, vol. 31, no. 5, Apr. 2020. [Online]. Available: https://doi.org/10.1002/ett.3961

[56] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions," 2009. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf

[57] "Cisco Visual Networking Index: Forecast and Trends, 2017 - 2022," White Paper. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf

[58] ETSI NFV, "Network Functions Virtualisation (NFV); Management and Orchestration," December 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[59] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, thirdquarter 2017.

[60] C. X. Mavromoustakis, J. M. Batalla, G. Mastorakis, E. Markakis, and E. Pallis, "Socially oriented edge computing for energy awareness in iot architectures," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 139–145, July 2018.

[61] Y. Nikoloudakis, E. Markakis, G. Alexiou, S. Bourazani, G. Mastorakis, E. Pallis, I. Politis, C. Skianis, and C. Mavromoustakis, "Edge caching architecture for media delivery over p2p networks," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Sep. 2018, pp. 1–5.

[62] C. X. Mavromoustakis, G. Mastorakis, and J. Mongay Batalla, "A mobile edge computing model enabling efficient computation offload-aware energy conservation," *IEEE Access*, vol. 7, pp. 102 295–102 303, 2019.

[63] H. Liao, Z. Zhou, S. Mumtaz, and J. Rodriguez, “Robust task offloading for IoT Fog computing under information asymmetry and information uncertainty,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[64] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, “Joint task offloading and resource allocation for delay-sensitive fog networks,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–7.

[65] X. He, R. Jin, and H. Dai, “Deep PDS-Learning for Privacy-Aware Offloading in MEC-Enabled IoT,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4547–4555, June 2019.

[66] Z. Ning, P. Dong, X. Kong, and F. Xia, “A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, June 2019.

[67] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, “On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[68] S. Shahzadi, M. Iqbal, T. Dagiuklas, and Z. U. Qayyum, “Multi-access edge computing: open issues, challenges and future perspectives,” *Journal of Cloud Computing*, vol. 6, no. 1, p. 30, Dec 2017.

[69] L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta, “Joint optimization of service function chaining and resource allocation in network function virtualization,” *IEEE Access*, vol. 4, pp. 8084–8094, 2016.

[70] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, “Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements,” in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec 2015, pp. 1–7.

[71] L. T. Bolivar, C. Tselios, D. Mellado Area, and G. Tsolis, “On the deployment of an open-source, 5g-aware evaluation testbed,” in *2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, March 2018, pp. 51–58.

[72] J. Haavisto, M. Arif, L. Lovén, T. Leppänen, and J. Riekki, “Open-source rans in practice: an over-the-air deployment for 5g mec,” in *2019 European Conference on Networks and Communications (EuCNC)*, June 2019, pp. 495–500.

[73] M. Z. Khan, S. Harous, S. U. Hassan, M. U. Ghani Khan, R. Iqbal, and S. Mumtaz, “Deep Unified Model For Face Recognition Based on Convolution Neural Network and Edge Computing,” *IEEE Access*, vol. 7, pp. 72 622–72 633, 2019.

[74] A. C. Baktir, A. Ozgovde, and C. Ersoy, “How Can Edge Computing Benefit From Software-Defined Networking: A Survey, Use Cases, and Future Directions,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2359–2391, Fourthquarter 2017.

[75] “Linux LXD Containers.” [Online]. Available: linuxcontainers.org/lxd

[76] A. C. Soong and R. Vannithamby, *5G Verticals*. John Wiley and Sons, Ltd, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119514848.ch1

[77] D. T. Nguyen, C. Pham, K. K. Nguyen, and M. Cheriet, “Placement and chaining for run-time iot service deployment in edge-cloud,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 459–472, 2020.

[78] 5G PPP Architecture Working Group, “View on 5G architecture,” February 2020. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-5G-Architecture-White-Paper_final.pdff

[79] ETSI, “Multi-access edge computing (MEC); framework and reference architecture,” January 2019, group Specification. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/mec/001_099/003/02.01.01_60/gs_mec003v020101p.pdf

[80] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3098–3130, 2018.

[81] M. Alizadeh and T. Edsall, "On the data path performance of leaf-spine datacenter fabrics," in *2013 IEEE 21st Annual Symposium on High-Performance Interconnects*, 2013, pp. 71–74.

[82] T. Doukoglou, V. Gezerlis, K. Trichias, N. Kostopoulos, N. Vrakas, M. Bougioukos, and R. Legouable, "Vertical industries requirements analysis targeted kpis for advanced 5g trials," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 95–100.

[83] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and qos-aware path allocation and vnf placement for service function chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2019.

[84] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 643–655, 2019.

[85] X. Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau, "Online VNF scaling in Datacenters," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016, pp. 140–147.

[86] G. Moualla, T. Turletti, and D. Saucez, "Online robust placement of service chains for large data center topologies," *IEEE Access*, vol. 7, pp. 60 150–60 162, 2019.

[87] D. Harutyunyan, N. Shahriar, R. Boutaba, and R. Riggio, "Latency and mobility-aware service function chain placement in 5g networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[88] T. Gao, X. Li, Y. Wu, W. Zou, S. Huang, M. Tornatore, and B. Mukherjee, "Cost-efficient vnf placement and scheduling in public cloud networks," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4946–4959, 2020.

[89] I. Sarrigiannis, L. M. Contreras, K. Ramantas, A. Antonopoulos, and C. Verikoukis, "Fog-enabled scalable c-v2x architecture for distributed 5g and beyond applications," *IEEE Network*, vol. 34, no. 5, pp. 120–126, 2020.

[90] A. Solano and L. M. Contreras, "Information exchange to support multi-domain slice service provision for 5g/nfv," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 773–778.

[91] Y. Wang, P. Lu, W. Lu, and Z. Zhu, "Cost-efficient virtual network function graph (vnfg) provisioning in multidomain elastic optical networks," *Journal of Lightwave Technology*, vol. 35, no. 13, pp. 2712–2723, 2017.

[92] "CPLEX optimizer." [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[93] Cisco, "White Paper Cisco data center spine-and-leaf architecture:design overview," January 2020. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.pdf

[94] ETSI, "Multi-access edge computing (MEC); framework and reference architecture," February 2018, group Specification. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/mec/001_099/017/01.01.01_60/gr_mec017v010101p.pdf

[95] K. Antevski, C. J. Bernardos, L. Cominardi, A. de la Oliva, and A. Mourad, "On the integration of nfv and mec technologies: architecture analysis and benefits for edge robotics," *Computer Networks*, vol. 175, p. 107274, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620300797

[96] ETSI, "5G; 5G system; application function (AF) event exposure service," August 2020, technical Specification. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129517/16.01.00_60/ts_129517v160100p.pdf

[97] T. Toukabri, "5G-TRANSFORMER system design and techno-economic analysis," 2019. [Online]. Available: https:

//ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5c9d303ba&appId=PPGMS

[98] M. R. Garey and D. S. Johnson, *Computers intractability: A guide to theory NP-completeness*. Bell Telephone Laboratories, 1979.

[99] S. Martello and P. Toth, *"Knapsack problems," Algorithms and Computer Implementations.* Wiley, 1990.

[100] 5GAA Automotive Association, "White Paper C-V2X use cases, methodology, examples, and service level requirements," June 2019. [Online]. Available: https://5gaa.org/wp-content/uploads/2019/07/5GAA_191906_WP_CV2X_UCs_v1-3-1.pdf

[101] M. A. U. Gawas, "An overview on evolution of mobile wireless communication networks: 1g-6g," *International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC)*, vol. 3, no. 5, pp. 3130–3133, 2015.

[102] 5G Slicing Association, "Categories and service levels of network slicing white paper," March 2020. [Online]. Available: https://www-file.huawei.com/-/media/corporate/pdf/news/categories-slice--white-paper-en.pdf?la=en

[103] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, 2017.

[104] K. M. S. Huq, J. Rodriguez, and I. E. Otung, "3d network modeling for thz-enabled ultra-fast dense networks: A 6g perspective," *IEEE Communications Standards Magazine*, vol. 5, no. 2, pp. 84–90, 2021.

[105] A. Al-Kinani, C.-X. Wang, L. Zhou, and W. Zhang, "Optical wireless communication channel measurements and models," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 1939–1962, 2018.

Appendix A

List of Acronyms

1G	first generation
2G	second generation
3G	third generation
3GPP	3rd generation partnership project
4G	fourth generation
5G	fifth generation
5GC NF	5G core network function
6G	sixth generation
AaaSFC	application-as-a-SFC
AF	application function
AI	artificial intelligence
AMF	access and mobility management function
AP	access points
API	application programming interfaces
AS3	simple storage service
AUSF	authentication server function
AWS	Amazon web services
B5G	beyond 5G networks
BSS	business support system
CAGR	compound annual growth rate
CAPEX	capital expenditure
CNF	cloud-native network functions
CPM	cost per minute
CPU	central processing unit
CUPS	control and user plane
C-V2X	cellular V2X
DB	database
DN	data network
E2E	end-to-end
EC2	elastic compute cloud

eMBB	enhanced mobile broadband
EMS	element management system
EPC	evolved packet core
ETSI	European telecommunications standards institute
EU	European Union
FCloud	foreign cloud
FMEC	foreign MEC
gNB	5G base station
GSMA	global system for mobile communications association
HP	high priority
HSS	home subscriber service
IaaS	infrastructure-as-a-service
ILP	integer linear programming
IoT	Internet of things
IoV	Internet of vehicles
ISG	industry specification group
ITS	intelligent transportation system
k8s	kubernetes
LB	load balancer
LCloud	local cloud
LCM	lifecycle management
LCVNF	latency-critical VNF
LMEC	local MEC
VT	virtualization technology
LP	low priority
LTE	long term evolution
LTVNF	latency-tolerant VNF
MANO	management and orchestration
MEC	multi-access edge computing
ML	machine learning
MME	mobility management entity
mMTC	massive machine type communication
MNO	mobile network operator
NEF	network exposure function
NFV	network function virtualization
NFVI	NFV infrastructure
NFVO	NFV orchestrator
NIC	network interface card
NR	new radio
NRF	network repository function
NSA	non-standalone
NSD	network service descriptor
NSSF	network slice selection function
ONAP	open network automation platform
OPC	operator platform concept
OPEX	operational expenditure
OS	operating system

OSM	**open source MANO**
OSS	**operations support system**
OWC	**optical wireless communication**
PCF	**policy control function**
PCRF	**policy and charging rules function**
PDU	**protocol data unit**
PGW	**packet gateway**
PNF	**physical network function**
PoP	**point of presence**
QoE	**quality of experience**
QoS	**quality of service**
RAM	**random access memory**
RAN	**radio access network**
SA	**standalone**
SBA	**service based architecture**
SDN	**software defined networking**
SFC	**service function chain**
SGW	**serving gateway**
SLA	**service level agreements**
SMF	**session manager function**
SoA	**state-of-the-art**
SSD	**solid state disk**
U.S.	**United States**
UAV	**unmanned aerial vehicle**
UDM	**unified data manager**
UE	**user equipment**
UHD	**ultra high definition**
UPF	**user plane function**
URLLC	**ultra-reliable low-latency communication**
V2I	**vehicle-to-infrastructure**
V2N	**vehicle-to-network**
V2V	**vehicle-to-vehicle**
V2X	**vehicle-to-everything**
vCPU	**virtual CPU**
VIM	**virtualized infrastructure manager**
VM	**virtual machine**
VNF	**virtual network functions**
VNFD	**VNF descriptor**
VNFFG	**VNF forwarding graph**
VNFM	**VNF manager**
vPGW-C	**virtualized PGW control plane**
vPGW-U	**virtualized PGW user plane**
vRAM	**virtual RAM**
vSGW-C	**virtualized SGW control plane**
vSGW-U	**virtualized SGW user plane**
VT	**virtualization technology**