Universidade Estadual de Campinas
Instituto de Computação

Diogo Machado Gonçalves

Dynamic Network Slicing for User Mobility Support in 5G Networks

Fatiamento Dinâmico de Redes para Usuários Móveis em Redes 5G

CAMPINAS
2024

Diogo Machado Gonçalves

Dynamic Network Slicing for User Mobility Support in 5G Networks

Fatiamento Dinâmico de Redes para Usuários Móveis em Redes 5G

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Edmundo Roberto Mauro Madeira
Co-supervisor/Coorientador: Prof. Dr. Luiz Fernando Bittencourt

Este exemplar corresponde à versão final da Tese defendida por Diogo Machado Gonçalves e orientada pelo Prof. Dr. Edmundo Roberto Mauro Madeira.

CAMPINAS
2024

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

G586m

Gonçalves, Diogo Machado, 1993-
Migração proativa de máquinas virtuais em computação em névoa para usuários móveis / Diogo Machado Gonçalves. – Campinas, SP : [s.n.], 2018.

Orientador: Edmundo Roberto Mauro Madeira.
Coorientador: Luiz Fernando Bittencourt.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Computação em nuvem. 2. Máquinas virtuais. 3. Internet das coisas. 4. Dispositivos móveis. 5. Recursos de redes de computadores. I. Madeira, Edmundo Roberto Mauro, 1958-. II. Bittencourt, Luiz Fernando, 1981-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Proactive virtual machine migration in fog computing environments for mobile users
Palavras-chave em inglês:
Cloud computing
Virtual machines
Internet of things
Mobile devices
Computer network resources
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Edmundo Roberto Mauro Madeira [Orientador]
Alfredo Goldman Vel Lejbman
Leandro Aparecido Villas
Data de defesa: 10-10-2018
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Diogo Machado Gonçalves

Dynamic Network Slicing for User Mobility Support in 5G Networks

Fatiamento Dinâmico de Redes para Usuários Móveis em Redes 5G

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira
  IC/UNICAMP
- Prof. Dr. Alfredo Goldman vel Lejbman
  IME/USP
- Prof. Dr. Rafael Pasquini
  CETEC/UFU
- Prof. Dr. Magnos Martinello
  CT/UFES
- Prof. Dr. Carlos Alberto Astudillo Trujillo
  IC/UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 03 de outubro de 2024

Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools, it is about how we use them and what we find out when we do.
(Edsger Dijkstra)

Acknowledgements

This journey is likely the one I am most proud of. During this time, I had the best family, pets, advisers, labmates, housemates, colleagues, and friends I could ever wish for. I visited some of the most remarkable places in the world and even lived in a few of them. However, this PhD was also a journey within myself, taking me to places I never imagined existed. From -38 degrees in Waterloo to 38 degrees in Rio, from sedentary days working from home to running a half marathon in Pisa, every place, every person, and every experience during this period has shaped who I am today. In this section, I would like to cite some of them.

From my home in Rio, I thank my love, Thalita, for inspiring me to be the best person I can be and for introducing me to the greatest dog to ever walk on this Earth. My life is better with you both.

From my hometown, I thank my parents, Luisa and Osmar, my whole family, and my friend Anderson for supporting me throughout my life, even though I am not as close as I wish.

From Campinas, I thank Professors Edmundo Madeira and Luiz Bittencourt for being the best advisors any student could wish for. You have both made me a better researcher and person. I also want to thank my friends from LRC, the quietest and most alcohol-free lab at UNICAMP. My PhD journey would have been far harder without you all!

From Canada, I thank Professors Raouf Boutaba and Nashid Shahriar from the University of Waterloo for hosting me in their lab. There, I discovered why the University of Waterloo is among the best in the world. My thanks to Lucy for welcoming me into her home and offering me great English lessons.

From Wales, UK, I thank Professors Omer Rana and Ioan Petri for hosting me at Cardiff University. I'm grateful for the valuable research discussions and meetings we shared. Thanks also to my Egyptian friend and office mate Amin for our diverse conversations and to housemates Steve and Shane for the enjoyable chats in the kitchen. A special thanks to Shane for running with me through Cardiff's rainy days.

From Italy, I thank Professors Carlo Puliafito and Enzo Mingozzi for our long-time research partnership. It has been a pleasure working with you.

Finally, I want to thank everyone who contributed directly or indirectly to this work in some way.

This work is part of the INCT of the Future Internet for Smart Cities (CNPq 465446/2014-0, CAPES 88887.136422/2017-00, and FAPESP 2014/50937-1). This work was funded by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, grant 2021/05253-0 by São Paulo Research Foundation (FAPESP), and grant 382766/2022-9 by CNPq.

Resumo

O Fatiamento de Rede vem se mostrando uma das peças chave para o gerenciamento dos recursos de infraestruturas modernas de rede como o 5G. A partir da virtualização dos recursos físicos da rede, diversas Redes Virtuais (Fatias de Rede) podem ser criadas sobre uma mesma infraestrutura a fim de servir, sob medida, grupos distintos de usuários. Esses usuários, por sua vez, acessam através das Redes Virtuais, serviços e aplicações presentes na infraestrutura. Além dos requisitos de armazenamento, processamento e rede (ex., latência, largura de banda e confiabilidade) intrínsecos a cada aplicação, características dos usuários, como mobilidade, acrescentam complexidade para esse cenário. Nesse contexto, não somente as aplicações acessadas pelos usuários devem ser alocadas em locais estratégicos, como também as Redes Virtuais devem ser construídas a fim de prover a conexão necessária entre esses usuários e suas aplicações. Dada a dinamicidade desse cenário, a infraestrutura de rede deve ser capaz de se reconfigurar de forma eficiente à medida que a demanda por recursos se modificar ao longo do tempo. Esta tese de doutorado visa apresentar novas soluções para o *Problema do Alocação Dinâmica de Redes Virtuais* com o objetivo de atender Usuários Móveis em uma infraestrutura de Névoa/5G. Em especial, este trabalho desenvolveu soluções para dois problemas de otimização relacionados à alocação dinâmica de recursos presentes nessa arquitetura: a reconfiguração de Redes Virtuais e a Migração de Serviços. A contribuição deste trabalho inclui: 1) Um simulador desenvolvido para permitir a validação das soluções propostas; 2) Uma avaliação do comportamento das redes virtuais em face à dinamicidade de demanda provocada pelos usuários móveis; 3) Um modelo matemático que calcula o custo, medido em tempo, para a alocação e reconfiguração de uma rede virtual; 4) Uma solução heurística para otimizar o processo de reconfiguração das redes virtuais. O simulador desenvolvido está disponível como software livre e tem sido adotado como ambiente de validação por parte da comunidade científica. Os estudos realizados indicam que a realocação dinâmica de recursos em redes virtuais se mostra como uma solução para a infraestrutura de rede se adequar às variações de demanda causadas por usuários móveis. Entretanto, a frequência em que a infraestrutura é capaz de executar tais reconfigurações é determinante para o sucesso de tal solução. Reconfigurações mais rápidas tendem a redistribuir mais rapidamente os recursos da rede entre as fatias de rede de acordo com a demanda dos seus usuários. Uma redistribuição de recursos mais eficiente provê níveis de latência mais baixos e estáveis aos usuários atendidos pelas fatias de rede.

Abstract

Network slicing has been a key supporting technology for resource management in modern computer networks like 5G. Based on the virtualisation of physical resources, virtual networks (slices) are created over one common physical infrastructure. In that context, each user group is served by one virtual network tailored to reach their requirements. Furthermore, the users access services and applications present in the infrastructure through these slices. The main challenge of 5G networks is simultaneously accommodating applications and services with a wide range of diverse and sometimes conflicting requirements while serving users with different characteristics, such as mobility. To accomplish such objectives, services should be placed in strategic sites, and the slices should provide the required connection between users and their services. Furthermore, both services and slices should be able to adjust to fit demand variations caused by user mobility quickly. Based on the described scenario, this thesis developed new resource management mechanisms for the *dynamic network slicing allocation problem* to support mobile users in 5G/fog computing infrastructures. In special, this work proposes dynamic network slicing mechanisms as a supporting technology for improving the process of placing and migrating services in fog. The contributions of this work include: 1) a network simulator has been developed for evaluating the proposed solutions; 2) one evaluation of the performance of dynamic network slicing in user mobility support scenarios; 3) a mathematical model which describes the overhead in time perspective of network slices reconfiguration; and 4) two heuristic solutions for improving the network slice reconfiguration process. The simulator is available as free software and has been adopted as the validation environment by the scientific community. The results of this work indicate that the dynamic resource allocation in network slices can be helpful for network infrastructures in dealing with demand variations triggered by mobile users. However, the frequency of those reconfigurations is crucial to the success of such a solution. Faster reconfigurations tend to quicker converge their resources to a stable level that is suitable for their users' demand. Slices that quickly balance their resources provide lower and more stable latency to their users.

List of Figures

List of Tables

List of Acronyms

AP	Access Point
API	Application Programming Interface
CU	Central Unit
CN	Core Network
DCSP	Data Centre Service Providers
DPD	Deployment Process Delay
DU	Distributed Unit
E2E	End-to-End
eMBB	Enhanced Mobile Broadband
FMC	Follow-Me Cloud
GUI	Graphical User Interface
gNB	Next Generation Nodes B
IoT	Internet of Things
ITS	Intelligent Transportation System
KPI	Key Performance Indicator
LuST	Luxembourg SUMO Traffic database
MANO	NFV Management and Orchestration system
MCC	Mobile Cloud Computing
MIPS	Million Instructions Per Second
ML	Machine Learning
mMTC	Massive machine-type communication
NF	Network Function
NFV	Network Function Virtualisation
NOP	Network Operator

NS	Network Slice
NSaaS	Network Slice as a Service
NSDT	Network Slicing Deployment Time
OPD	Onboarding Process Delay
PN	Physical Network
QoD	Quality of Decision
QoS	Quality of Service
RAN	Radio Access Network
ROD	Runtime Orchestration Delay
RRU	Radio Remote Unit
RSU	Roadside Unit
SC	Service Customer
SDN	Software-Defined Network
SFC	Service Function Chain
SP	Service Provider
SUMO	Simulation of Urban Mobility
UE	User Equipment
UML	Unified Modelling Language
URLLC	Ultra-Reliable Low-latency Communications
V2E	Vehicle-to-Everything
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad Hoc Network
VCC	Vehicular Cloud Computing
VFC	Vehicular Fog Computing
VISP	Virtualisation Infrastructure Service Providers
VM	Virtual Machine

Contents

Chapter 1

Introduction

Since the First Generation of Mobile communications, several improvements were developed in each new version aiming to supply the increasing users' requirements over time, as illustrated in Figure 1.1. The services provided by these networks started with voice calls in the first generation, moved to the introduction of text messages in 2G, access to mail and web pages in 3G, and significant improvements in voice and video transmissions in 4G. However, the fifth generation is the one that aims to be a breakthrough in the telecommunication area. The first generation of mobile communication systems to support different vertical industries, such as connected vehicles, e-health, and Industry 4.0 manufacturing is the 5G. Unlike the previous generations that focused on voice, text, and video communications, 5G support for machine-to-machine requests from Internet-of-Things (IoT) devices, including other requirements [147]. Currently in development, 6G and beyond mobile communication systems aim to extend 5G by improving coverage, data rate and latency, to cite a few [36]. The next mobile communication system will enable digital twins, virtual reality and augmented reality, and aircraft and overseas applications [159].

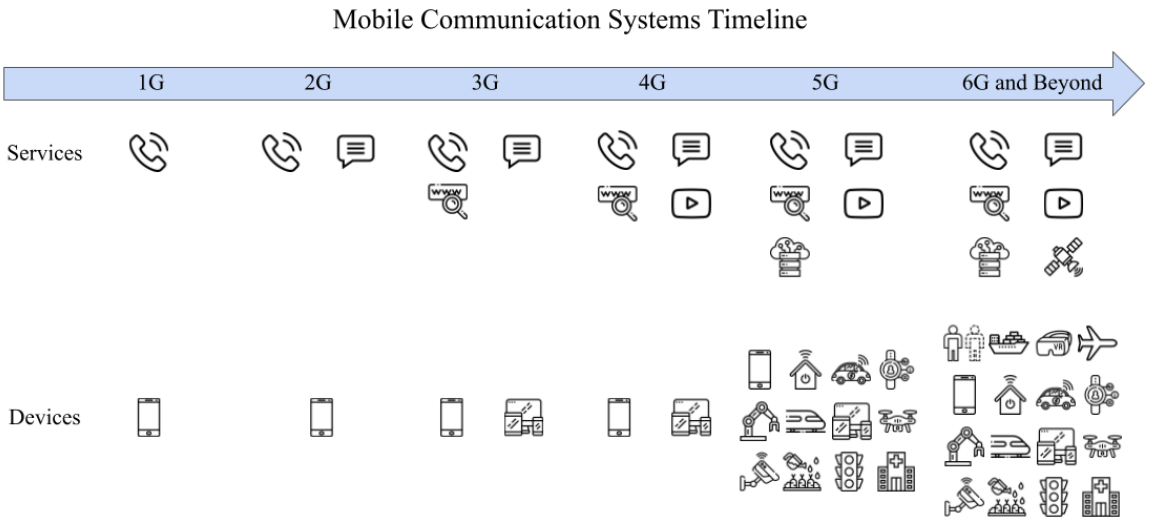


Figure 1.1: Evolution of Mobile Communication Systems from 1G to 6G and beyond.

In this context, modern mobile communication systems like 5G and beyond have the focus on, but it is not limited to, supporting three big generic categories [141], each one

with its particular requirements, as illustrated in Figure 1.2:

- Enhanced mobile broadband (eMBB), which requires improvements in the next-generation broadband access such as higher throughput and spectrum efficiency, amplified coverage area, and enhanced mobility support;
- Massive machine-type communications (mMTCs) are related to the huge number of IoT devices, like wearables, smart home devices, and rural sensors, which can be connected simultaneously in a single base station. In such a scenario, it is required efficient connections in terms of energy consumption in an environment with high connection density;
- Ultra-reliable low-latency communications (URLLCs), the most critical ones, are related to real-time applications, such as remote surgery, autonomous navigation, and manufacturing. These applications require highly reliable and available End-to-End (E2E) connections with ultra-low latencies.

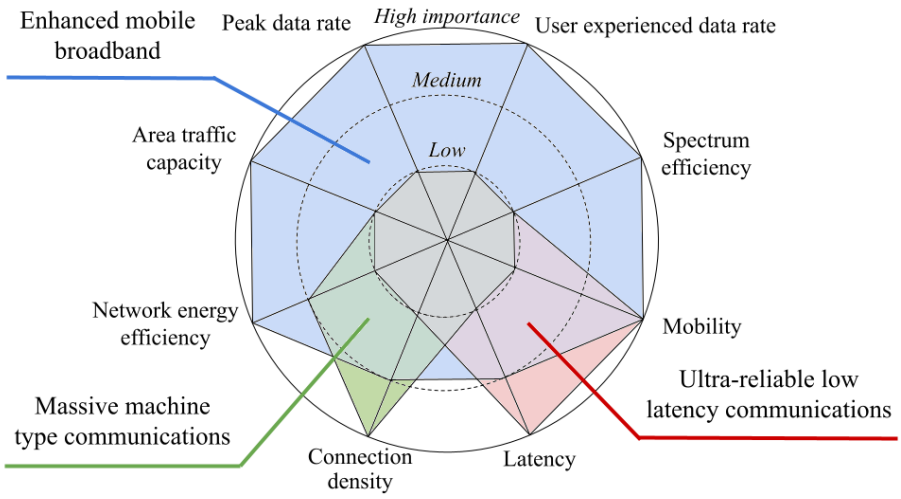


Figure 1.2: Levels of requirements by different user groups. Source: Adapted from [141]

Aiming to support such different users, the main challenge of those networks is simultaneously accommodating applications and services with a wide range of diverse and sometimes conflicting requirements. To achieve that purpose, 5G selects different supporting technologies to compose its ecosystem. 5G is composed of a fog computing architecture that provides processing and storage resources at the edge of the network and LTE and optical networks to connect users and nodes. However, the traditional logical architecture might not efficiently manage the network resources and successfully fulfil all the user requirements.

A possible way to let networks address such requirements is through Software-Defined Networking (SDN), Network Functions Virtualisation (NFV), and Network Slicing (NS) [163]. Specifically, SDN and NFV [19] decouple Network Functions (NF), *e.g.*, firewalls and routing, from the underlying network hardware, implementing them as a virtualised environment (virtual machines or containers). These technologies provide great flexibility, as they

allow the creation, deletion, and movement of network functions from one hardware device to another on-demand and in a timely way. Network slicing leverages SDN and NFV to create different logical networks on top of the same physical or virtual network. Each logical network (*i.e.*, network slice) is tailored to fulfil the requirements of a particular group of users or applications. Tailored network slicing and end-to-end slicing, *i.e.*, from the wireless connection at the edge to the processing server at the core of the network, will still guide the development of network slicing for 6G and beyond networks [159]. In special, resource allocation is "one crucial technical issue" for network slicing development in 6G [25].

In special, this thesis focuses on mobile users and how network slicing can manage edge network resources to serve them. Facing a dynamic scenario, in special, because of the demand variations due to the attendance of mobile users, the network needs to quickly and continuously orchestrate its resources to serve its users. Some of these challenges are investigated in this thesis. The scope of this work is discussed in the following sections of this chapter, which includes presenting the motivation in Section 1.1, the problem statement and objectives in Section 1.2, contributions in Section 1.3, and the scientific production in Section 1.4.

1.1 Motivation

Smart Cities solutions aim to improve the quality of life and the well-being of citizens by minimising urban problems [88]. However, integrating the physical world and decision-making systems is challenging [137]. Machine learning (ML) algorithms and Cloud Computing infrastructures have been playing a successful role as smart city supporters [116, 10]. Network slicing is also pointed as an enabler for smart cities applications [126, 163]. Among various smart city solutions, Intelligent Transportation Systems (ITS), which avoid traffic jams or connected video surveillance systems that improve the city's security, are some examples, to cite a few [156].

In that context, data collected from users is sent to cloud data centres to be processed. In those centralised servers, ML algorithms act as decision-makers. However, due to the high volume of data collected from users, cloud computing-based solutions face a bottleneck performance in terms of latency, throughput, or both [21]. Also, communication costs increase when data is sent from end users to the core of the network. Aiming to mitigate that scenario, processing data closer to end users, *i.e.*, at the edge of the network, improves real-time data processing, reduces transmission latency and avoids data transfer in the core of the network [21]. On the other hand, edge devices have lower computing power than cloud servers. In both architectures, edge and cloud resources are usually handled as a virtualised environment that shares its physical resources in the form of virtual machines (VM) or containers.

Many other applications in smart cities scenarios can take advantage of edge computing resources. However, this thesis focuses on those scenarios with high processing and low latency requirements and user mobility characteristics. Vehicles driven on streets as part of an Intelligent Transportation System is one of those smart city scenarios. In

special, that scenario is pointed as an example of a technical challenge for network slice management in the development of beyond 5G networks [126]. That context is assumed as an example of the applicability of the evaluated scenario of this thesis.

Intelligent Transportation Systems join information technology and transportation and transit systems to provide a better resource use of different transportation actors [168]. Typically, these include conventional and self-driven cars, trains, bicycles, and road traffic lights, to cite a few. In that context, a vehicle equipped with video cameras, distance sensors and communication technologies could collect data from different sources and send it to be processed in edge servers near roads. Large-scale systems can use both edge and cloud resources to process and manage ITS [61]. Also, vehicular networks, that can include the Vehicle-to-Vehicle (V2V) and Vehicle-to-Everything (V2E) communications increase the diversity of ITS applications. Figure 1.3 illustrates that scenario in which vehicles, traffic lights, roadside servers, edge and cloud computing, and other actors and resources are integrated in a ITS.

Minimising traffic jams and pollution and improving drivers' and passengers' safety are part of the urban mobility plan of every city. In recent years, different solutions have been proposed for ITS in the context of edge-assisted infrastructures [168, 55]. Those solutions include, among others, collision avoidance and optimal route planning [18], traffic light detection [27] and object detection [47]. Pattern recognition tasks could be used in the perception of road signs, detection of pedestrians, vehicle detection in the surrounding area, and recognition of the actions of the driver and other actors, such as braking, steering, accelerating, lane changing, or crossing lanes [168]. Those examples of applications require reliable and low-latency network infrastructures.

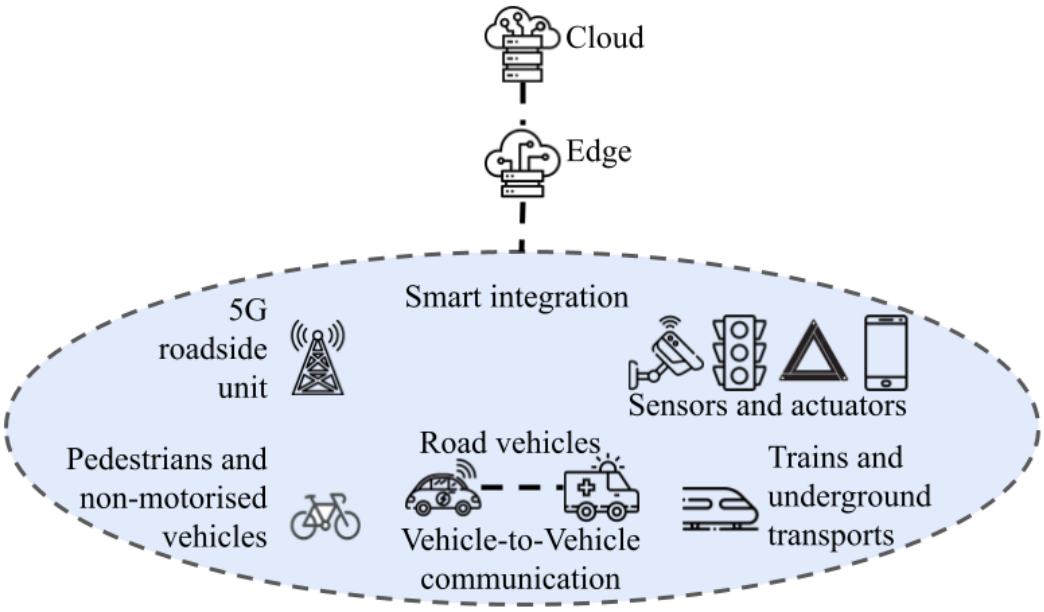


Figure 1.3: Intelligent Transportation System and its components in an edge-cloud architecture. Adapted from [55]

Ground vehicles in a ITS, such as cars, present a high mobility pattern and need remote computing resources to process robust applications. In addition, due to nature of

the user mobility, intense traffic variation is identified in this scenario, which increase the challenges of the network resource management [126, 73, 135, 6, 142, 144]. In particular, "the resources allocated to network slices must conform to the dynamics of network demands, which are unpredictable and vary with time and user mobility" [134]. In the face of such a dynamic scenario, *i.e.*, high user mobility and traffic variation, the network infrastructure needs to efficiently and dynamically manage its resources to avoid resource underutilisation and Quality of Service (QoS) degradation scenarios. Based on that challenge, mobility-based network slicing and on-demand mobility management are pointed as current and future research directions [126, 73, 135, 6, 142]. In this context, network resource management provided by dynamic network slicing could improve the performance of applications for mobile users while improving resource utilisation. That hypotheses is investigated in this thesis. The problem statement and the objectives of this work are discussed in the following section.

1.2 Problem Statement

Although the strength characteristics of network slicing, the slice orchestration required to achieve such benefits is a complex case [19]. In such a scenario, some problems remain unsolved. Some of these challenges have been investigated in this thesis. This work focuses on the *dynamic slicing allocation problem*. In special, this work proposes the use of dynamic network slicing as a supporting technology for improving solutions for the service placement and migration problem in the context of user mobility support in 5G/fog computing infrastructures. The relevance of edge resource management and user mobility support[164, 76, 175], and network slice management in edge[73, 132, 15, 157] remain applicable from 5G to 6G and beyond. In what concerns dynamic network slicing, "it is evident that there is a consensus pointing towards the superiority of dynamic network slicing as the preferred method for enabling agile and efficient network management in modern communication systems" [73]. A simulation environment is also required to validate these proposed solutions. A brief discussion about these challenges and the research questions that are investigated in this thesis is presented below.

1. *Resource allocation*: One of the main challenges in network virtualisation, known as Virtual Network Embedding Problem [56], is properly assigning the physical resources (node and links) to a virtual network. In the context of 5G infrastructures, the orchestration and management of network slices need to deal with different customers with specific characteristics and requirements. Furthermore, the slice instances may need to be reshaped to serve the new requirements due to spatial and temporal demand variations over time [134]. That resource allocation may be in a static or dynamic way [54], in which each approach introduces a trade-off in terms of performance and computing overhead. Among other resource management approaches in 5G, "dynamic resource allocation is the worthiest one to research" [91]. In special, this thesis aims to answer the following research question: *Is dynamic slicing a suitable solution to deal with traffic variation triggered by mobile users in edge computing environments?* Further details about that scenario are presented in

section 2.2.2;

2. *Resource sharing*: Due to the flexibility of network virtualisation, different isolation levels can be defined for the network resources. A network slice can be fully or partly isolated from other slice instances [11]. The isolation can be defined as physical and virtual resources (nodes sharing computing and storage resources and links sharing bandwidth and radio spectrum) [16], Network Functions, or multi-domain services. Besides, it is possible to combine those approaches. Facing that complex scenario, "although the dynamic resource sharing among slice tenants would make network resource utilisation more efficient, it calls for intelligent scheduling algorithms that will allocate resources among these slices" [19]. Although the importance of isolation for security, this work focuses on isolation aspects for resource utilisation improvements instead. Security concerns are out of the scope of this thesis. In this case, this thesis aims to answer the following research question: *What is the impact of resource sharing in the performance of network slicing for mobile users?* Further details are presented in section 2.2.1;

3. *Mobility management*: Among diverse customers served by 5G infrastructures, each one with its particular requirements, the network slicing orchestration needs to deal with not only its requirements but also with users' characteristics that increase the complexity of that scenario. Mobile devices like wearables, smartphones, or smart devices like vehicles and drones require continuous connectivity while moving among different sites. In such a scenario, seamless mobility management strategies need to be developed [126, 135, 19], which include, among other, seamless handover, service migration strategies, and dynamic resource allocation approaches to provide load balance. The following research question is answered in this work: *What is the impact of user mobility in network slicing management?*;

4. *Service migration*: The placement of Network Functions or services/applications has a strategic role in achieving latency requirements or load balance performance. Furthermore, due to demand variations caused by, for example, user mobility, the placement of NFs may need to be rebuilt to achieve the best configuration. In such a scenario, the NF/services are constantly reallocated in other nodes, which is known as the service migration process. In this scenario, service migration "is an efficient mechanism for the optimal placement of VNFs" [107]. Efficient migration strategies in terms of when, where, and how to perform such process play a key role in "improve a mobility support solution both in terms of performance and applicability" [122] in fog computing environments. However, "the problems concerning NFs placement within the slice, intra-slice management, and inter-slice management still need significant efforts in order to achieve and realise the effectiveness of the network slicing concept in 5G networks" [19]. One of the objectives of this work is to answer the following research question: *Can dynamic network slicing improve the performance of the service placement and migration process?* Further details about that scenario are presented in section 2.1.4;

5. *End-to-end vehicular-edge cloud*: Similarly to mobile phones, smart home devices,

and wearables, vehicles are gradually adopting more sensors, actuators, computing, storage and network resources. Intelligent Transportation Systems (ITS) can take advantage of those smarter devices to increase traffic rate and safety and decrease energy consumption and pollution on the roads [127, 40]. The data communication between vehicles form the Vehicular Ad-Hoc Networks (VANETs) [94] and, once vehicles' idle computing power could be shared among nearby vehicles, that pool of resources could form a Vehicular Cloud [115]. The aggregation of IoT devices into edge-cloud infrastructures is part of the IoT-Edge-Cloud continuum concept [110, 21]. In this context, "network slicing can provide dedicated resources for V2X services, ensuring reliable connectivity" [111]. Although network slicing has been evaluated in different contexts, there are not many works evaluating how network slice handles such dynamic scenarios. Efficient network slicing in vehicular environments is a research challenge [111, 6, 73, 14]. Based on that problem, the following research question is investigated in this work. *Can dynamic slicing be applied in vehicular-cloud environments?* Further details regarding this scenario are present in Section 5.3.

6. *Trade-off scenarios*: Latency, throughput, and reliability are three Key Performance Indicators (KPIs) in network contexts. In 5G architecture, fog nodes' storage and computing capabilities are also considered. The nodes and links involved in the slice's resource allocation process and the requirements of each user may present different values for each KPI. In that context, different priorities on some of these KPIs may be required, which impacts multi-criteria decision-making solutions. Furthermore, there are computational and time costs to execute resource reallocations, which directly impact the solutions' performance. Once a slice requests more resources, such resources should be reassigned as soon as possible. However, there is a trade-off between the reallocation cost and the benefits of that new redistribution [66, 45]. In that context, the time between the slice reconfiguration request and the moment in which slice is ready for operation should be considered in the performance evaluation. Based on that scenario, the following research question is answered in this thesis: *What is the required time to perform network slice operations such as resource allocation and slice reconfigurations and how it impacts the network performance?* Further details about trade-off scenarios and other key performance indicators are presented in Section 2.2.3 and Chapter 6.

7. *Environment validation*: One fundamental step in developing resource management mechanisms for network slicing is using reliable environments to validate such mechanisms. Although the advantages of the real testbed, economic limitations are presented as one of the main restrictions in adopting that environment. Based on that context, simulation and emulation tools raise as a feasible environment to validate these scenarios. Although there exist some simulators for evaluating the impact of resource management approaches in the performance of fog computing environments, none allow evaluation of scenarios that combine network slicing, service migration, and user mobility support. Based on that gap present in the literature, the following research question is answered in this work: *Is there a validation en-*

vironment to simulate network resource management scenarios which includes user mobility, network slicing and service migration?

The scope of this thesis includes all the cited challenges described above. The solutions required for this context need to simultaneously choose the fog nodes to place the users' services considering latency, throughput, and load balancing aspects while deciding when to perform service migrations due to demand variations caused by mobile users. In such a context, network slicing is used as a supporting technology for those migrations by improving the bandwidth availability. In addition to solutions to the service migration problem, the infrastructure should decide the shape of those slices, *i.e.*, which nodes and links compose each slice as well as the amount of resources allocated, while deciding the frequency in which one slice will be reshaped. The contributions of this thesis are discussed below.

1.3 Contributions

In the face of the challenges present in the 5G infrastructures introduced above, this thesis presents one evaluation of the performance of dynamic network slicing and develops new mechanisms to support dynamic network slicing as a solution for high dynamic scenarios, in particular, for user mobility support. *The main objective of this work is to propose resource management mechanisms to orchestrate dynamic network slices in 5G/fog computing architectures to support mobile users. In particular, this work proposes solutions combining dynamic slicing allocation, service placement, and migration to improve mobile users' QoS.* To achieve the main objective, five minor objectives combined compose this thesis. The contributions of this work are, as follows, the development of:

1. An heuristic solution which uses dynamic network slicing as a supporting technology to improve the service migration process;

2. A study of the dynamic network slicing as a supporting technology for user mobility support in fog/edge computing infrastructures and vehicular networks;

3. A mathematical model which describes the overhead/expected time to allocate and reconfigure the network slices;

4. Resource management mechanisms for dynamic slicing allocation, which include solutions for defining the slice shaping, resource allocation and reallocation, and resource sharing scope;

5. A simulator for slicing validation in fog computing-based infrastructures, including support to user mobility and service migration.

The evaluations of the developed mechanisms are based on three main metrics: latency, reliability/service outage and throughput. The evaluation of these metrics reflects how the infrastructure impacts the QoS provided to its users. The performance of the proposed

mechanisms was measured in terms of processing time, reflecting the scalability of the solutions.

In order to validate these mechanisms, an environment that supports the evaluated scenarios while ensuring reliability and scalability is required. Given that such a solution is only available in the literature for testbeds, one of the main objectives of this work is to provide such an environment for the academic community. The outcomes of this thesis in terms of peer-reviewed papers and software are discussed below.

1.4 Scientific Production

The contributions of this work are focused, but not limited to, on proposing solutions for dynamic network slicing to support user mobility scenarios in fog/edge infrastructures. During the development of this thesis, the proposed content, which includes mathematical models, simulation scenarios, algorithms, and software, was peer-reviewed and published in relevant journals and conferences. Those papers are listed as follows:

- C. Puliafito, D. M. Goncalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. Bittencourt. Mobfogsim: Simulation of Mobility and Migration for Fog Computing. Simulation Modelling Practice and Theory, 101:102062, 2020 [121];

- D. Gonçalves, C. Puliafito, E. Mingozzi, O. Rana, L. Bittencourt, and E. Madeira. Dynamic Network Slicing in Fog Computing for Mobile Users in Mobfogsim. In 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), pages 237–246. IEEE, 2020 [66];

- D. M. Gonçalves, L. F. Bittencourt, and E. R. M. Madeira. Fatiamento Dinâmico de Redes em Computaçao em Névoa para Usuários Móveis. In Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, pages 57–70. SBC, 2021. (In Portuguese) [67];

- D. M. Gonçalves, L. F. Bittencourt, and E. R. M. Madeira. Alocação de Fatias de Rede Fim-a-fim para Usuários Móveis Utilizando o Simulador Mobfogsim. In Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, pages 112–125. SBC, 2022 (In Portuguese) [68];

- D. M. Gonçalves, C. Puliafito, E. Mingozzi, L. F. Bittencourt, and E. R. M. Madeira. End-to-end Network Slicing in Vehicular Clouds Using the Mobfogsim Simulator. Ad Hoc Networks, page 103096, 2023 [70];

- D. M. Gonçalves, L. F. Bittencourt, and E. R. M. Madeira. Overhead and Performance of Dynamic Network Slice Allocation for Mobile Users. Future Generation Computer Systems, 2024 [69];

- D. M. Gonçalves, L. F. Bittencourt, and E. R. M. Madeira. MobFogSim: Simulation of Mobile Applications in Edge Computing Environments. Tutorial presented at the IEEE International Conference on Cloud Networking 2024[1].

Although the primary purpose of this thesis is to develop a new resource management mechanism for network slicing, some fundamental steps are needed before developing those new approaches. Based on that, the initial focus of this thesis was building a realistic, flexible, reliable, and scalable environment to validate the resource management mechanisms to be proposed.

In that context, the first step to building such an environment was the development of MobFogSim [121], an open-source simulator[2] derived from other fog simulators, MyiFogSim [92] and IFogSim [71]. MobFogSim enhances iFogSim by modelling user mobility and VM/container migration among fog nodes. These features make MobFogSim a comprehensive tool for evaluating fog computing networks where end-user devices are both fixed and mobile. More specifically, MobFogSim implements a *Migration Policy* that decides when to migrate a user's VM/container. This decision is based on parameters such as (i) the user's position, direction, and speed and (ii) the occurrence of a wireless handoff (*i.e.*, change of access point performed by the mobile device). Instead, the *Migration Strategy* decides where to migrate the service, *i.e.*, it chooses the next fog node to run the service. This selection may be based on the geographical or topological proximity of the new fog node to the mobile device, but other criteria can also be implemented. MobFogSim is also friendly with the output of others mobility simulators, *e.g.*, the Simulation of Urban MObility (SUMO) [20]. That feature enables simulations to be carried out using data from realistic mobility databases. Data from a real testbed was used to validate the simulator results. The validation of the simulator is further discussed in Section 4.7. That work is published in the 2020 edition of the Simulation Modelling Practice and Theory journal [121]. It is worth mentioning that Diogo M. Gonçalves and Carlo Puliafito share the first author role in that paper, as highlighted in that document.

MobFogSim was one of the first fog computing simulators with user mobility support to provide service migration. However, at that point, the simulator did not support some of the most promising resource management mechanisms present in the state of the art, especially the network slicing support. Based on that, the second step of the development of the simulator was adding support to dynamic network slicing. MobFogSim implements network slicing by orchestrating the distribution of network resources (topology and link bandwidth) among the slices. The slices share the same physical infrastructure in the simulator; however, each slice reserves a pool of resources. Resources reserved by a slice are dedicated to users within that slice. That work is present in the proceedings of the 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC) [66].

New features of MobFogSim in terms of the simulator's scalability and support to end-to-end slices, which include the management of Fog Nodes' processing and storage resources, were presented in the proceedings of the 2022 XV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos [68]. Finally, the device-to-device communi-

[1]https://cloudnet2024.ieee-cloudnet.org/tutorials

[2]https://github.com/diogomg/MobFogSim. Last accessed: April 10th, 2024.

cation was implemented in the MobFogSim and described in the 2023 edition of the Ad Hoc Networks journal [70]. Further details about the development of the MobFogSim simulator are described in Chapter 4.

An experimental evaluation of how dynamic network slicing impacts service migration to support mobile users in a fog environment was presented in [66]. Besides evaluating network slicing by considering different resource allocation approaches and levels of resource requirements and availability, the impact of that technology on user mobility scenarios still need to be further evaluated. Based on the simulation setup in [66], additional scenarios were made to evaluate the performance of network slicing using a realistic user mobility pattern. That work is presented in the 2021 XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos [67]. In this thesis, Chapter 5 presents the performance evaluation of dynamic network slicing. In special, it presents the evaluation of static and dynamic network slicing allocation from [66], the role of dynamic slicing in serving mobile users' requirements from [67], how slices with different resource requirements impact the resource management from [68] and, how vehicular networks can explore the benefits of dynamic slicing from [70].

Aiming to share the applicability of MobFogSim, one tutorial was conducted at the IEEE CloudNet 2024. The hands on tutorial introduced MobFogSim and covered its features to the conference audience.

Despite the benefits of network slicing, that approach might be sensitive to computing overhead and trade-off scenarios, which are not fully evaluated in the literature. The impact of network slice reconfiguration overhead on the overall network performance was evaluated. In [69], we presented a resource allocation model that calculates the required time to allocate a slice instance. Furthermore, based on those values, that work evaluated the impact of resource allocation delays in terms of bandwidth allocation, latency, and service outage in different dynamic slicing scenarios. That work is published in the 2024 version of the Future Generation Computer Systems Journal [69]. Chapter 6 further discusses that topic.

Additional works were done in the meantime of the development of the main objectives of this thesis. One work presented an analysis of a service migration approach based on mobility prediction in a fog computing scenario. That work showed how different accuracy levels of mobility prediction impact the performance of the service migration. It raised a discussion about the trade-off between that approach's benefits and computing overhead. That work [65] is present in the proceeding of the 2019 XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

In addition, one book chapter was developed discussing resource allocation approaches in multi-layer edge computing infrastructures. In particular, that document focuses on mechanisms for content distribution and user mobility support in smart city scenarios. That work [42] is part of the 2021 version of the Mobile Edge Computing Book.

Focusing on network slicing scenarios, the contributions of those works are *not* discussed in this thesis, although they indirectly were part of this research.

1.5 Document Organisation

The remaining of this document is presented as follows:

- Chapter 2 presents key concepts and their relation to this thesis, like 5G networks and their architecture, service migration and their requirements, and network slicing;
- Chapter 3 introduces the main related work in the state of the art regarding network slicing mechanisms and network simulators. That chapter also presents network simulators present in the literature and points out the strengths and weaknesses aspects of MobFogSim in the face of them;
- Chapter 4 introduces the MobFogSim architecture and its features, and the experiments proceeded to validate the simulator;
- Chapter 5 presents extensive experiments to evaluate the performance of network slicing for user mobility support under many scenarios. Those scenarios include dynamic vs static slice allocation, dynamic slicing for user mobility support, the trade-off present in the simultaneous allocation of slices with different priorities, and the performance of dynamic slicing in vehicular networks;
- Chapter 6 introduces the discussion about the overhead present in the dynamic slice allocation in terms of time and computing consumption;
- Chapter 7 presents the conclusion of the thesis and introduces the topics that remain open and which direction the future works can follow.

Chapter 2

Background

This chapter introduces the basic concepts of topics considered in this thesis. This chapter is organised as follows:

- Section 2.1 presents an overview of 5G networks, which includes their physical and logical architectures, and introduces the rule of service migration and network slicing on that architecture;
- Section 2.2 introduces the concept of network slicing in 5G networks, which includes network slicing isolation, resource sharing and resource allocation. Also, that section defines the key performance indicators used in this work to evaluate the proposed scenarios.

2.1 5G Networks

This section introduces the key concepts related to 5G Networks and their relation to this thesis. 5G and its technologies compose the base architecture in which the proposed solutions of this work are developed. This section presents the 5G physical and logical architectures in Subsection 2.1.1. Based on that architecture, supporting technologies that compose the 5G ecosystem is presented. Cloud and fog computing are discussed in Subsection 2.1.2, and wireless and wired communications are presented in Subsection 2.1.3. The 5G orchestration in terms of Network Functions and how it is related to service migration are presented in Subsection 2.1.4.

Unlike the previous generations, 5G aims to support not only human's devices but also to provide machine-to-machine Internet-of-Things communications. In this scenario, the telecommunication's users include mobile phones and wearables, a wide range of vehicles, smart home devices, and manufacturing machines. Currently, the requirements for mobile communication systems include even higher throughput, lower latency, more extensive coverage, higher user mobility and energy efficiency, and security, availability, and reliability.

In this context, 5G networks must manage a complex scenario to serve these different users with distinct requirements. That complexity is justified not only because of the customers but also by other actors, like providers and operators, who are part of this

ecosystem and sometimes have conflicting interests. In this context, the 5G ecosystem should provide an environment for these actors to cooperate and sometimes compete among themselves. Figure 2.1 illustrates the relationship between these actors in the 5G ecosystem.

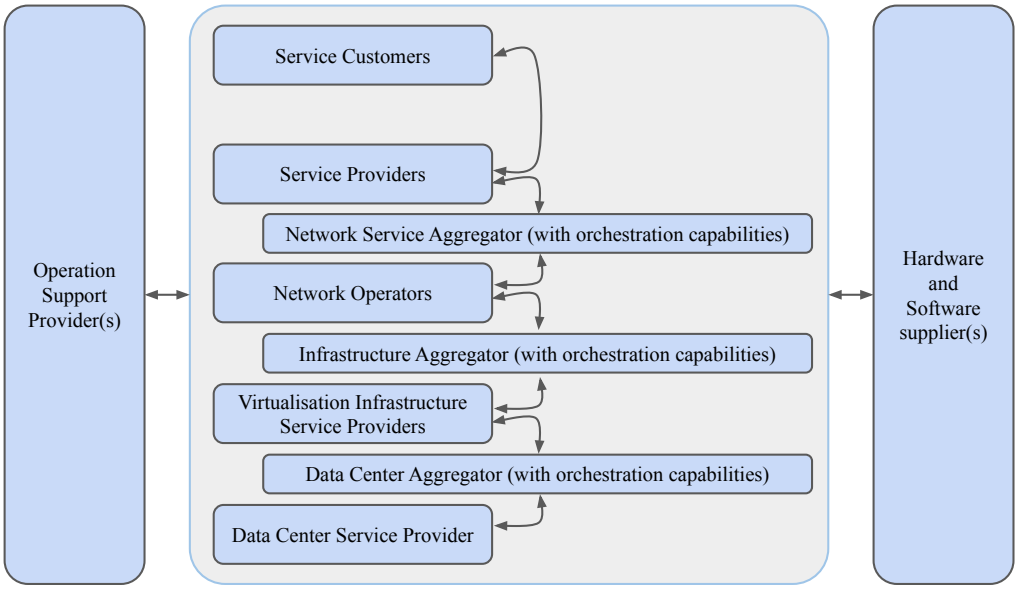


Figure 2.1: Relation between the different actors in the 5G ecosystem. Based on: [33]

.

Based on the definitions of [33], one of the main actors in this ecosystem is the Service Customer (SC), like vertical industries, which use the services of Service Providers (SP). The services offered by SPs can be in the form of traditional telecom services (known as Communication Service Providers), digital services like software support for chatbots, security cameras, and self-driven cars (known as Digital Service Providers), or in the form of network slices which contain the services supported by the SP (known as Network Slice as a Service (NSaaS) provider).

The resources used by the SP are offered by Data Centre Service Providers (DCSP) and Virtualisation Infrastructure Service Providers (VISP). These resources are orchestrated by a Network Operator (NOP). The DCSP provides computing resources in the form of physical data centres, while on top of them, VISP manages virtualised networks and computing resources from multiple technology domains. Based on these resources, Network Operators orchestrate multiple VISPs to offer network services to SP.

2.1.1 Architecture

The 3rd Generation Partnership Project (3GPP) [1] defines the 5G architecture as a set of one Core Network (CN)/5GC (5G Core) and one or more access networks, *e.g.*, a Radio Access Network (RAN), as illustrated in Figure 2.2. The CN is a highly virtualised environment composed of Network Functions (NFs) and services (further details in Section 2.1.4). At the same time, the RAN is responsible for providing an interface for the end-users to the Network Functions [33].

The RAN is composed of a set of next-generation nodes B (gNB) connected to each other and to the CN. The gNB connects User Equipments (UEs) and the Network Functions. Figure 2.3 describes the 5G architecture. NG, Xn, and F1 are logical interfaces that connect these components.

These Network Functions, deployed in the Central Unit (CU), Distributed Unit (DU), and Radio Remote Unit (RRU), can be placed through the network architecture from the edge to the cloud. The CU processes non-real-time protocols and services, and the DU processes physical level protocols and real-time services [141].

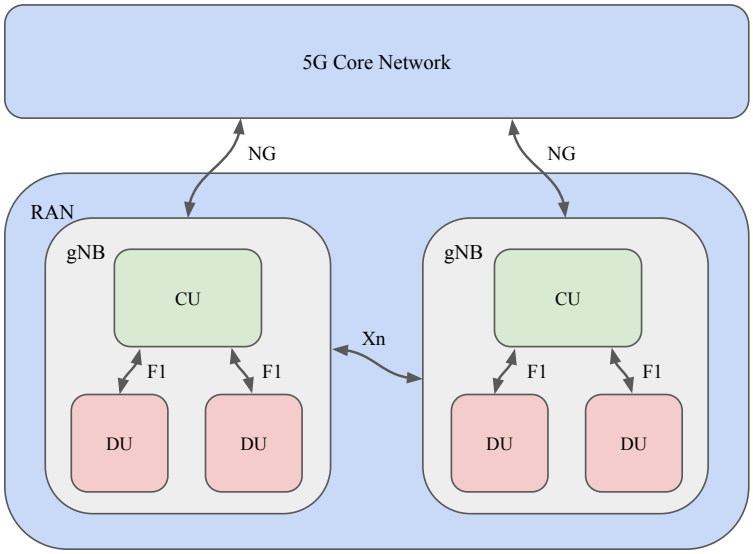


Figure 2.2: 3GPP logical architecture. Based on [1]

Due to the great flexibility provided by a highly virtualised environment, the Service Providers can change the placement of these Network Functions to meet specific requirements. Figure 2.3 illustrates a physical architecture for 5G networks with independent DU and CU placement.

In the 5G physical architecture, the connection between the edge's access points and the core network is called the transport network. In this scenario, sub-networks are defined based on the placement of services and network functions. In such a scenario, the fronthaul transport is defined between RRUs and DUs, the midhaul transport is defined between DUs and CUs, and the backhaul transport is between CUs and CNs.

2.1.2 Cloud and Fog Computing

In the 5G architecture, the orchestrations of the network resources are performed by network functions. Those NFs need computing and storage resources to execute their tasks properly. In this context, 5G can use some cloud and fog computing concepts to manage its nodes' resources. The placement of these cloud and fog nodes is illustrated in Figure 2.3.

The cloud computing paradigm aims to provide efficient and flexible resource management to its customers. The resources provided by the cloud, *e.g.*, storage and processing

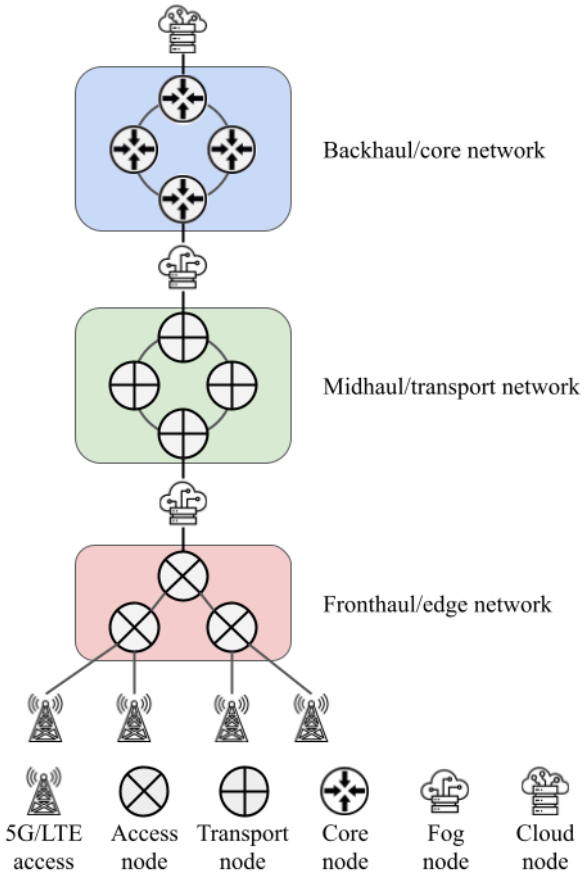


Figure 2.3: Example of a physical architecture for 5G networks.

resources, a development environment, or an application, are allocated or released on demand. In cloud architecture, access to resources is made by virtualisation, in which several Virtual Machines (VMs) can run over the same physical infrastructure. Cloud data centres are typically placed on the core of the network.

Regardless of the benefits of the cloud architecture, the fact that a high amount of computing resources are located far from the end-users may limit its ability to serve some applications, *e.g.*, delay-sensitive. In this context, the fog computing paradigm was raised as an extended cloud architecture that brings some resources closer to users. Fog infrastructures can consist of one or more layers, hierarchically organised, located between the users and the cloud. The number and composition of such layers depend on the specific application domain and its requirements. Typically, the highest layer is represented by the cloud. In contrast, the lowest fog layer is expected to be close (geographically) to the user at the edge of the network, *e.g.*, hosted at the first hop access point. In that architecture, nodes placed in higher layers are expected to present higher computing and storage capacity and higher latency. In contrast, due to the placement closer to the users, the fog nodes in lower layers are resource-poor devices with the lowest latency. In parallel to the 5G architecture, cloud nodes are placed at the core network and the fog nodes are placed along with the transport network and RAN, at the edge of the network.

2.1.3 Wireless and Wired Communication

In addition to cloud and fog nodes responsible for providing computing and storage resources to 5g architecture, one communication infrastructure must connect those entities. In that architecture, different network domains require specific communication characteristics. In 5G, the communication infrastructure can be defined in wired and wireless connections.

5G integrates different types of wireless technologies, such as the evolution of LTE, the 5G New Radio, and Wi-Fi-based technologies [33]. These wireless interfaces are embedded in the Radio Remote Unit (RRU), present in the RAN, as illustrated in Figure 2.3. The RRU connects the User Equipment (UE) and the 5G infrastructure. Each RRU has a radio coverage based on its wireless technology. Aiming to support user mobility, 5G implements a smooth transition for users moving from an RRU coverage to another. That process is named handoff.

On the other side of the 5G architecture, the core and transport networks concentrate on the data coming from the RRUs. These network links, illustrated as the NG, Xn, and F1 interfaces in Figure 2.2, require higher throughput capacity and availability, reliability, and energy efficiency guarantees. The transmission of the expected amount of data would hardly be reached with wireless technology. In this case, elastic optical technologies have been used in most cases, particularly because of their capability to be programmed and controlled according to service level requirements.

Due to virtualisation technologies, both wireless and wired resources are provided in the form of virtualised resources. These communication resources can be allocated based on the network topology and available throughput.

2.1.4 Network Functions and Service Migration

Unlike previous generations, 5G's breakthrough is strongly related to its flexibility provided by resource virtualisation, further explained in Section 2.2. These virtualisation technologies enable the 5G to disassociate a Network Function (NF) previously restricted to dedicated hardware to present them as Virtualised Functions. Based on the definition of [1], the virtualisation enables an NF to be "implemented either as a network element on a dedicated hardware, as a software instance running on a dedicated hardware, or as a virtualised function instantiated on an appropriate platform, *e.g.*, on a cloud infrastructure".

The 5G architecture is defined as service-based, in which the resource management is strongly related to system calls to these Network Functions. Based on [1], the 5G architecture consists of a set of 20 NFs. Authentication Server Function (AUSF), Access and Mobility Management Function (AMF), Unstructured Data Storage Function (UDSF), Network Slice Specific Authentication and Authorisation Function (NSSAAF), and Network Slice Selection Function (NSSF) are some examples of 5G Network Functions.

A service offered by a service provider uses the resources from the 5G infrastructure through the system interfaces provided by these NF. Once each NF performs a specific purpose, a service may require a set of network functions. In most cases, network function

interactions need to be made in a specific order. That order is defined as Service Function Chaining (SFC).

Aiming to optimise an SFC, the placement of these NF or an entire application may change over time due to workload changes or other situations such as load balance, availability concerns, or energy saving. In opportunistic scenarios, additional instances of these NFs or applications can be allocated to improve certain conditions, *e.g.*, load balance. Furthermore, some instances can be migrated from one node to another one to be, for instance, closer to their users. That process is known as service migration.

Due to the virtualisation of the 5G resources, further detailed in Section 2.2, these services are provided in the form of Virtual Machines (VM) or containers. Both virtual platforms provide an environment with processing, storage, and network resources for the service. Despite similar architectures, containers are more lightweight than the VM regarding memory consumption and, therefore, data to transmit during migration.

Different transmission techniques can be applied in the migration process. The two main migration techniques are called complete (cold) and live (pre-copy and post-copy) migrations. The Cold migration stops the container before saving and transmitting its state; it then resumes the container at the destination only when all the state has been transferred. Therefore, the time interval during which the container is unavailable to its users (*i.e.*, downtime) coincides with the total migration time, which is the overall time required to complete the migration. Post-copy migration involves stopping the container only to save and transfer a small portion of the overall state. That characteristic leads the migrations to a reduced downtime. The container then resumes at its destination, and while it is running, the rest of its state gets migrated. Post-copy is a "live" migration technique because the container remains active most of the time during the migration process.

For convenience, the remainder of this document will refer to the application running over the VM as *service*. That application can be either an application provided by a Service Provider or a network function. This rule will be applied unless a direct contrast is required.

2.2 Network Slicing

This section introduces the concept of network slicing, its definition, and related technologies. Network slicing has a key role in the development of resource management mechanisms for 5G infrastructures. In this work, network slicing is used to dynamically manage 5G's resources in order to serve mobile users. The remaining of this section presents the relation of network slice to 5G physical and logical architectures. Relevant aspects of network slicing are presented in the Subsections. Network slicing isolation is discussed in 2.2.1, and slicing resource allocation is discussed in 2.2.2. Finally, 2.2.3 introduces the key performance indicators defined by 3GPP to evaluate network slicing architectures.

In traditional networks, network functions are embedded in the hardware. Under such conditions, changes in demand result in physical changes to the network infrastructure.

There are situations in which this kind of infrastructure cannot always fulfil application requirements. One example is that of mobile users who present different characteristics of mobility (*e.g.*, speed of movement) and have different Quality of Service requirements (*e.g.*, latency, throughput). In this context, resource virtualisation supported by NFV and SDN [19] arises as a key supporting technology. NFV decouples Network Functions from the underlying hardware and presents them as virtual functions. SDN, instead, introduces the concept of programmable networks, which makes it possible to control network resources through Application Programming Interfaces (APIs). The combination of NFV and SDN enables the concept of network slicing.

Based on that scenario, network slicing has been introduced to improve the flexibility of networks. Network slices use the virtualisation of network functions to create different logical networks on top of the same physical or virtual Network [163]. Each slice can be created on-demand with different network characteristics according to its users' requirements. Some entities working on 5G development define the network slice as follows:

- Internet Engineering Task Force (IETF): To enable creating (End-to-End) partitioned network infrastructure that may include the user equipment, access/core transport networks, edge, and central data centre resources to provide differentiated connectivity behaviours to fulfil the requirements of distinct services, applications, and customers [96].

- 3rd Generation Partnership Project (3GPP): Slicing enables the operator to deploy multiple, more or less independent End-to-End networks potentially with the same infrastructure [33].

- Global System for Mobile Communications (GSMA): A network slice is an independent End-to-End logical network that runs on shared physical infrastructure, capable of providing an agreed service quality [16].

- Next Generation Mobile Networks Alliance (NGMN): A set of Network Functions and resources to run these Network Functions, forming a complete instantiated logical network to meet specific network characteristics required by the service instance (s) [11].

As illustrated in Figure 2.4, based on SDN and SFV virtualisation, network slicing creates different logical networks over the same physical network. Each logical network (*i.e.*, a network slice) is created to serve a group of users with specific needs. Network slices can cover all the network domains (*i.e.*, end-to-end slice) or only some of them (radio access network (RAN), transport network (TN), or core network (CN)) [26]. Furthermore, slices can present different isolation levels: some of the logical resources or Network Functions of a slice can be shared with other slices or actors outside the domain. Network slices enable better control of network resources and provide users with a virtual network tailored to their requirements and characteristics. The orchestration of those virtual resources, *i.e.*, network functions and slices, which includes allocation, placement, chaining, scaling, migration, fault recovery, and security are made by the NFV Management and Orchestration (MANO) system [97].

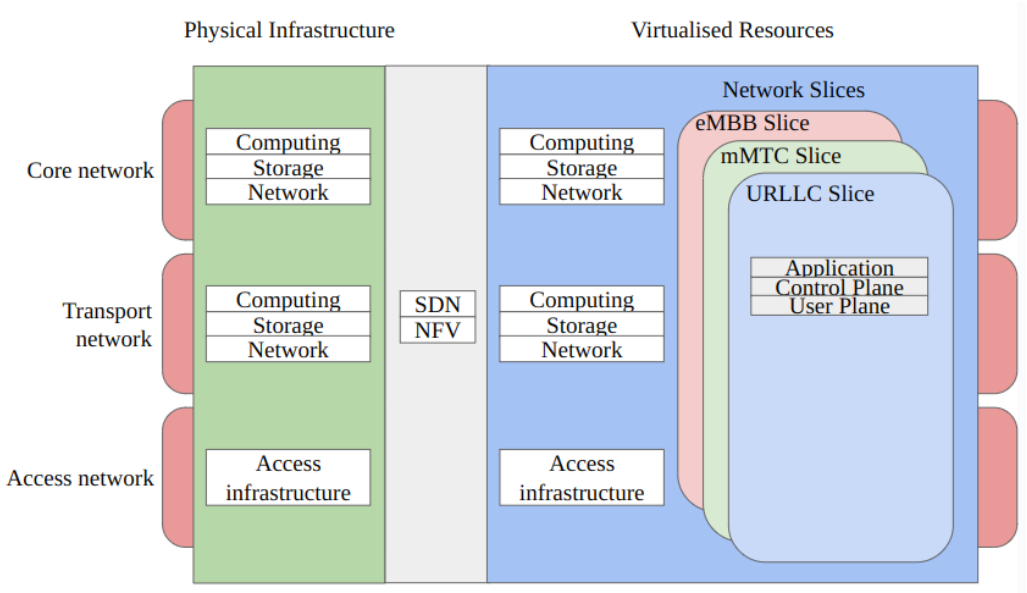


Figure 2.4: Logical architecture of a slice-based 5G network.

Figure 2.5 illustrates the concept of network slicing. As shown, there is a physical network infrastructure serving three logical networks. Each of them addresses the requirements of a specific category of applications. Hence, the first logical network is dedicated to Enhanced Mobile Broadband (eMBB) applications requiring high throughput. The second slice supports Machine-type Communications (mMTCs), which require efficient connections in terms of energy consumption in an environment with high connection density. The third and last logical network serves Ultra-Reliable Low-Latency Communications (URLLCs) applications, which require high reliability and availability with ultra-low latencies.

The technology enables network slicing to be transparent to business customers for whom 5G networks allow connectivity and data processing tailored to specific business requirements. It creates the illusion of the slice tenant operating its dedicated physical network. Each slice can be customised for different services and/or businesses. Service guarantees are associated with resource attributes such as throughput, packet loss, latency, reliability, network bandwidth/burst, or other bit rates and security. Redundancy and reliability are provided by the infrastructure to improve overall Quality of Experience (QoE). Concepts such as fog computing allow opportunistic placement of services to meet stringent requirements of low latency, high bandwidth applications, or both.

Mobile network operators could deploy a single network slice type that satisfies the needs of multiple verticals. Furthermore, multiple network slices of different types can be packaged as a single product to serve one user's bundle with multiple and diverse requirements. For instance, a vehicle may simultaneously need a high bandwidth slice for entertainment services and an ultra-reliable slice for telemetry-assisted driving. Providing an independently deployed E2E network slice for each vertical customer is unnecessary because this is not their fundamental requirement.

One network slice could serve many vertical customers simultaneously. The number of slices should be designed based on the capability of the underlying deployed infrastructure

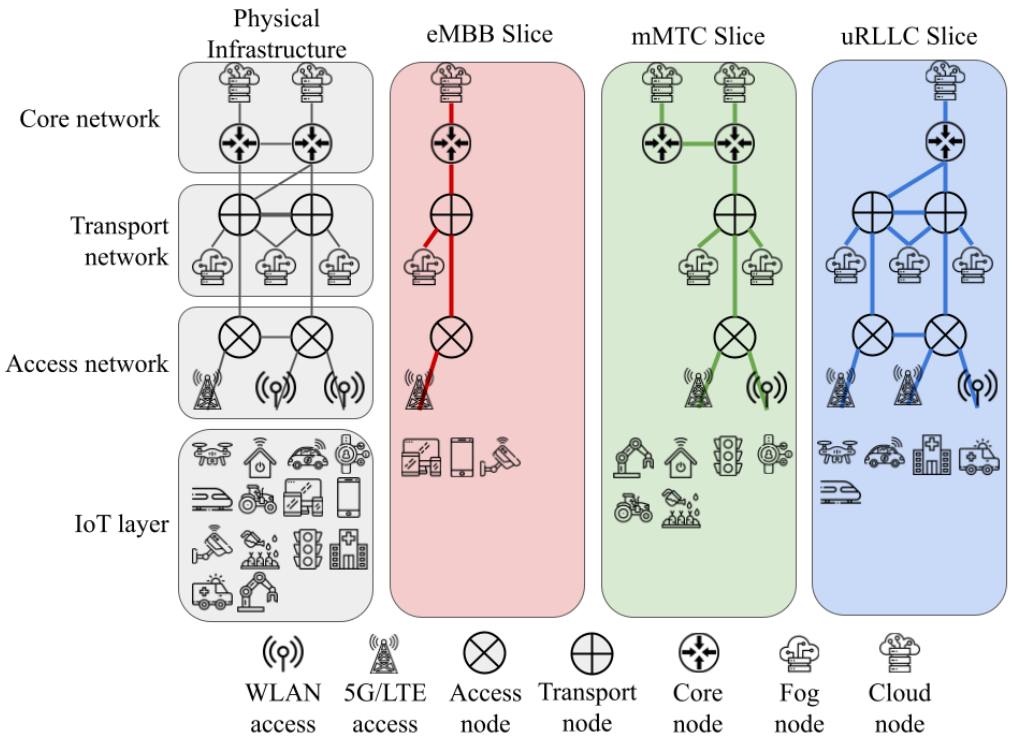


Figure 2.5: Visual representation of the network slicing concept.

and the performance, functional and operational requirements of business customers.

2.2.1 Isolation and Resource Sharing

One of the most relevant aspects of defining network slicing is the level of isolation. Due to the flexibility provided by virtualisation, network slices can provide many scenarios of different combinations of dedicated or shared resources and Network Functions. A network slice Instance may be wholly or partly, logically and/or physically, isolated from another network slice instance [11]. A network slice comprises dedicated and/or shared resources, *e.g.*, in terms of processing, power, storage, and bandwidth, and has isolation from the other network slices [16].

Slicing network isolation can be classified based on different aspects. Based on [16], the slicing isolation can be described as operational and network-level isolation. In operational isolation, vertical customers can have dedicated monitoring, control, configuration, or even full operational capability of the network slice. On the other hand, in network-level isolation, vertical customers do not share Network Functions or resources with the other customers. Network level isolation also has different sub-categories, for instance, shared RAN, but the isolated core, or isolated RAN and shared core. Figure 2.6 illustrates that scenario in the 5G architecture.

In addition to the definition of levels of slicing isolation based on physical resources, network slice isolation can also be defined by Network Functions or services. In the first definition, being a network slice instance composed of Network Functions, these NF can be dedicated to that slice or shared with others. Figure 2.7 illustrates the scenario of resource isolation based on Network Functions. In the Figure, NF1, NF2, and NF3 are

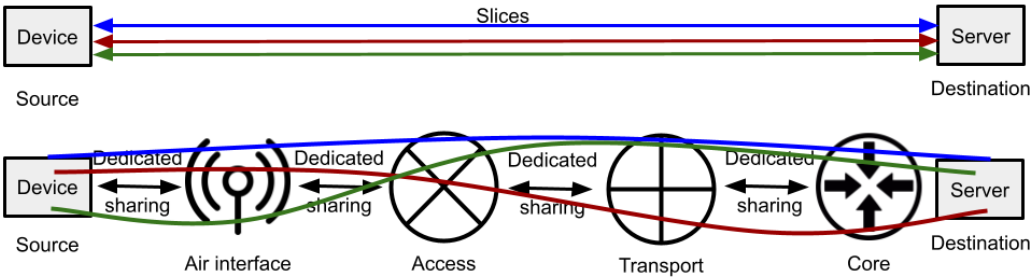


Figure 2.6: Levels of resource isolation based on physical resources.

dedicated to network slice 1 and NF8, NF9, NF10, and NF11 are dedicated to network slice 2; however, NF4, NF5, NF6, and NF7 are shared between network slice 1 and network slice 2.

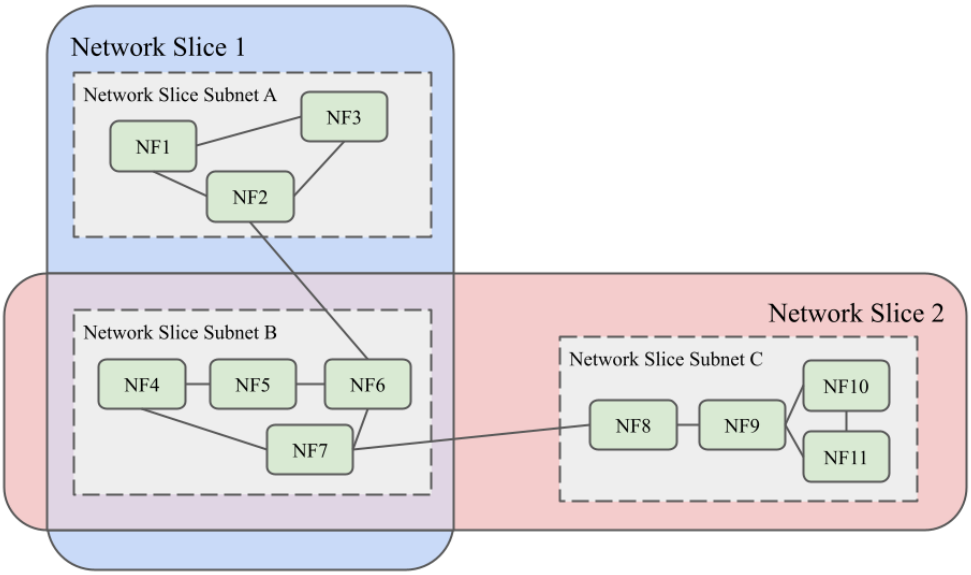


Figure 2.7: Levels of resource isolation based on Network Functions.

In the context of slicing isolation based on Service Customers, the whole network slice can be dedicated to one or more customers. Figure 2.8 presents the visual concept of slicing isolation based on services. The figure shows that Slice 1 is shared with Service 1 and 2, Service 2 uses both Slice 1 and Slice 2 and Slice 3 is dedicated to Service 3.

In addition, these three points of view of slicing isolation (based on physical resources, network functions, and services) can also be combined, *e.g.*, two network slices can share the same physical infrastructure but be composed of dedicated network functions or even two services share one slice instance composed only of dedicated NF.

2.2.2 Resource Allocation

In addition to resource isolation, resource allocation policy presents a relevant aspect in the network slicing environment. Although the ability to select tailored resources to compose an allocation of an NS instance, dynamic demand of services, *e.g.*, the frequency of request, can dramatically change resource requirements.

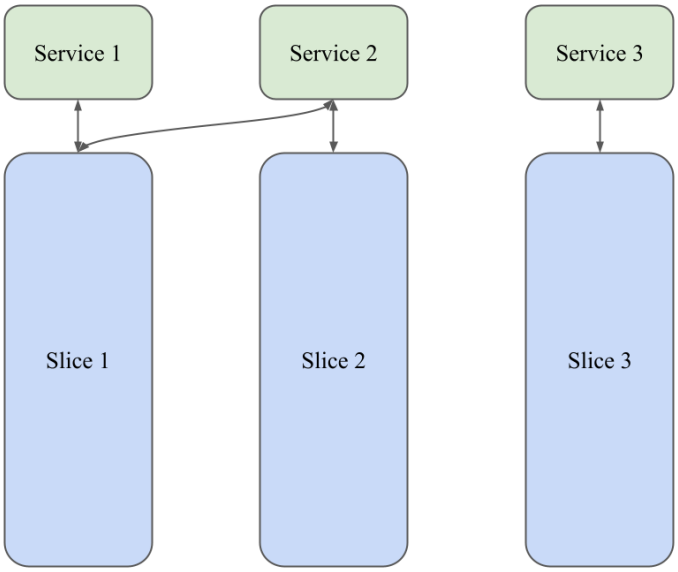


Figure 2.8: Levels of resource isolation based on services.

Network resources can be assigned to each slice either statically or dynamically. According to its demand, a static network slice receives a fixed portion of network resources. This number of resources remains allocated to that slice until the slice is deactivated. Further resources cannot be assigned to a static slice at runtime. On the contrary, dynamic network slicing allows a slice to acquire and release resources according to dynamic demand, thus enabling scaling-up and scaling-down [54]. Figure 2.9 illustrates the slice's resource reallocation based on temporal demand variations. Resource reallocation can be made in terms of cloud resources (computing and storage) and network resources (link bandwidth and topology), which are represented in the figure as generic Physical Network (PN) resources. In Figure, two VNs serve two different Service Customers (SC) with different temporal variations of the resource requirements. In that scenario, the VN can be reshaped on-demand based on SC's demand.

The network infrastructure needs to deal with different scenarios of resource change. Changes in user density due to user mobility, issues on reliability and availability aspects (nodes or links outages), or variable demand intrinsic to users' services may lead to changes in required resources over time.

The capability of service providers to orchestrate dynamic slices and adjust the allocated resources according to service demands may introduce computing overhead in the process of predicting such changes or reallocating the slice's resources.

2.2.3 Key Performance Indicators

Key Performance Indicators (KPIs) are a set of quantifiable data that can explain or suggest insights for effective network operations over time, such as planning, performance analysis and optimisation. This section discusses Key Performance Indicators applied in network slicing from different perspectives. Also, this section highlights the KPIs and

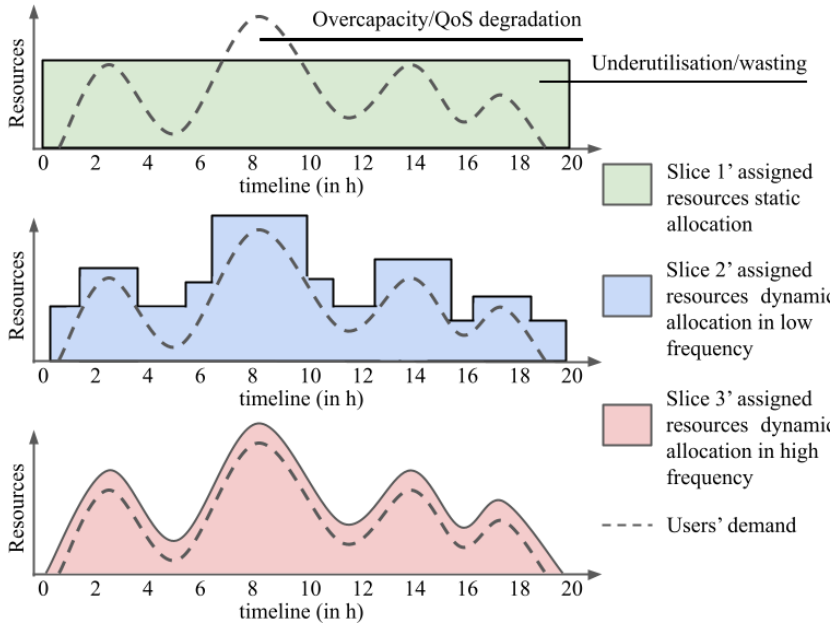


Figure 2.9: Slice's dynamic resource allocation due to demand variations.

metrics assumed in this work as part of the performance evaluation of the network slicing management.

Different actors in the 5G ecosystem have different objectives and KPIs. Some of those points of view are presented in this section. The performance evaluation of network slicing from the network operator perspective can be defined in 3 levels, as specified in 3GPP TS 28.554 [2] and 3GPP TS 28.552 [3]. Those technical specifications define KPIs for the NS monitoring in terms of network function/entity level, network slice subnet level and network slice level. The evaluation of the network function level is based on QoS aspects. The network slice subnet is evaluated based on its set of network functions and entities. Finally, the KPIs for network slicing can be derived from its network slice subnets and network function levels.

Quality of Service (QoS) and Quality of Experience (QoE) are metrics that aim to quantify the degree of satisfaction or annoyance of the users [99]. As there is no standard defined for 5G networks regarding those metrics, it becomes subjective to evaluate it. Based on that, 3GPP [2] defines some network KPIs to evaluate the performance of the network and the services. The last versions of those technical specifications highlight the following KPI categories: accessibility, integrity, utilisation, retainability, mobility, energy efficiency, and reliability.

From accessibility KPIs, 3GPP describes the number of registered subscribers of the network slices in terms of, *e.g.*, mean and maximum. In integrity KPIs, packet delay and packet loss rate define the required time to transmit data over the network from one point to another and the rate of unsuccessful transmission, respectively. Bandwidth defines the transmission capacity of the network link, while throughput indicates the actual amount of data transmitted through that upstream or downstream channel.

Utilisation is another KPI category defined by 3GPP regarding resource management and monitoring. The technical specification defines some metrics for virtualised resource

usage measurement. A weighted average represents the mean usage of VNFs virtualised resources for a specific period. The resources assumed are processing, memory, and disk). The equipment vendor defines the period and the weights applied in that KPI. Regarding user mobility management, 3GPP defines, among others, the number of requests and successful handovers between base stations, *i.e.* a mobile device switching its connection from one base station to another one. 3GPP defines Energy Efficiency as the performance of that network instance, *e.g.*, data volume, divided by the energy consumption.

Availability and reliability KPIs are also defined. 3GPP [2] defines reliability as the "percentage value of the packets successfully delivered to a given system entity within the time constraint required by the targeted service out of all the packets transmitted". In this work, service outage is the KPI assumed to evaluate the reliability rate of the network.

Finally, another point of view for network slicing evaluation is defined by the performance of network slice operations executed by VNFs Management and Orchestrations Systems (MANO). In addition to resource allocation and distribution, mobility management and network KPIs, the performance of slice operations can also impact the users' QoE. Although the result of dynamic operations performed by slices, such as resource allocation, can improve and ensure the KPIs previously discussed in this section, the execution of those operations is not instantaneous. The delay between the slice operation request and its readiness can sometimes result in a service outage. Due to its relevance, the overhead of slice operations is further investigated in this thesis in Chapter 6.

In this work, the performance of network slicing is measured mainly by the following KPIs: service latency, service throughput, network slice throughput, number of service migrations, duration of service migrations/service outages, link usage, number of slice reconfigurations and the slice outage due the reconfiguration process. Also, the performance and the overhead in terms of computing and time costs are investigated. However, different works in the literature might assume different KPIs to evaluate the performance of network slicing solutions. This section introduced some basic concepts regarding 5G networks, network virtualisation, service placement and migration, and resource allocation in the context of network slicing. In the following section, different state-of-art solutions found in the literature regarding those topics are discussed. Furthermore, the solutions proposed in this thesis are compared to those works.

Chapter 3

Related Work

This chapter introduces the main related work in the state of the art. Several works propose solutions based on Network Slicing for different problems related to resource management in network infrastructures. In the literature, there are works that propose network slicing resource management mechanisms considering user mobility support in its solutions [114, 106, 29, 154, 91, 35]. The placement and migration of services, VNFs, and SFC are also studied by researches [91, 161, 155, 119, 53, 167]. However, most of those works do not assume any values for the computing overhead for the resource allocation nor evaluate the impact of that overhead in their scenarios. On the other hand, there are works that focus on the evaluation of the overhead of network operations in the context of network slicing but do not consider user mobility, service placement or migration in that context [35, 155, 119, 53, 167, 28, 80, 5, 63]. Only a restrict group of works consider overhead costs of the slice operation in a user mobility scenario [7, 17, 109, 142].

This work is one of the few works in the literature that evaluates the *overhead of dynamic slicing allocation* due to traffic variations in user mobility scenarios. To the best of our knowledge, this work is the only one to proposes dynamic bandwidth slicing allocation to improves the service migration. Although the existence of works considering the impact of slice reconfiguration, the metric assumed as overhead is mostly related to monetary or computing costs. This work is the only one to consider the impact in the processing delay to commit one slice reconfiguration. One mathematical model that describes the expected time to proceed with one slice reconfiguration is proposed. This chapter further describes those cited papers related to the context of this thesis and highlights its contributions. A set of recent systematic review surveys published from 2022 to 2024 were used as the core source of research for related works. The selected surveys include resource management of network slicing [126, 37, 75, 151, 45, 44, 165, 134, 43], resource management of edge computing environments [176, 138, 41], and mobility management of slices [135, 111, 6, 73, 14]. Additional articles were included in this section due to their relevance.

The content of this chapter is based on four topics:

- Section 3.1 introduces a set of works that propose solutions for resource management problems of network slicing;
- Section 3.2 presents works related to mobility and service migration;

- Section 3.3 presents the state of art works that evaluate the overhead in network slicing allocation;
- Section 3.4 presents an overview of network simulators and how MobFogSim are compared to them.

Table 3.1 summarises the papers presented in Sections 3.1, 3.2, and 3.3. In addition, Table 3.2 summarises the papers of Section 3.4.

3.1 Network Slicing

Based on virtualisation and programmable capabilities, the network's flexibility improved by network slicing is one of the strengths of that technology. In this context, several works propose solutions based on network slicing for different problems related to resource management in a network infrastructure. This section introduces some articles in the literature related to the context of this work. Table 3.1 summarises these works.

[49] proposes a Deep Reinforcement Learning approach for network slicing placement in order to improve resource usage. A resource consumption reward is considered in the proposed model. The execution time of the algorithm is also evaluated. Although it is considered in the evaluation, the cited work does not further evaluate the impact of dynamic network allocations over time. Also, user mobility is not considered.

The authors in [114] propose one Deep Reinforcement Learning (DRL) solution for optimising the resource utilisation at the edge network. That work focuses on a set of edge nodes as resource offers. Similar to our work, the proposed end-to-end slices serve vehicular and smart city users with latency constraints in a dynamic environment. The proposed solution performs admission control tasks, accepting or denying new users and service requests. Although mobility is considered in slicing management, that work assumes a density ratio of high-utility applications to create different scenarios. On the other hand, this work uses individual user mobility patterns from a dataset to set the dynamism of the scenarios. Furthermore, a deep analysis of the overhead costs of the allocations is not evaluated.

In [106] proposes a semi-decentralised network slicing framework. Based on a deep reinforcement learning approach, a vehicle in each slice performs autonomous radio resource selection for V2V transmission. The proposed approach enables the vehicles' slice configuration to be self-configured and self-optimised under the supervision of the road-side infrastructure. The cited work considers user mobility in its evaluation. However, that work does not present service migration and anan overhead evaluation of the slice operationsare.

The work presented in [29] proposes a heuristic for dynamic bandwidth allocation in the context of user mobility support. The proposed solution is based on Deep Reinforcement Learning and aims to improve its mobile users' Quality of Experience (QoE). Similarly to our work, the cited work considers a dynamic bandwidth allocation triggered by mobile users. However, a further overhead evaluation is not considered in that work.

The authors in [154] take advantage of user mobility to predict the average demand on slices. The proposed approach predicts the number of received requests for all available

slices. Based on that prediction demand, a bandwidth slicing reservation scheme based on Neural Networks is used to serve mobile users. Despite considering user mobility to manage bandwidth in the RAN, the cited work does not provide a further evaluation regarding the computing and time overhead of the reconfiguration process. Also, realistic user mobility patterns are not considered as triggers for the traffic load variation.

In [35], a reinforcement learning algorithm is proposed to allocate resources in vehicular networks. The proposed solution dynamically allocates spectrum and computing resources in the RAN based on a deep reinforcement learning approach. The slices serve mobile users, which are assumed to be vehicles. The work considers the slice reconfiguration cost in its reconfiguration process for both spectrum and computing allocation. Despite considering user mobility, no realistic mobility pattern or a large-scale map is assumed.

3.2 Service Migration

In addition to network slicing topics, the placement and migration of network functions, services, and applications also play a critical function in resource management in fog computing infrastructures. This section presents a set of relevant works that investigates that topic. It is worth highlighting the shortage of work that uses network slicing to improve the service migration process.

The work in [91] proposes one bandwidth allocation based on reinforcement learning. That work discriminates the link traffic into migration and non-migration traffic, which can be understood as two different slices competing for bandwidth. The authors aim to maximise the number of migrations. That work validates its scenarios in MatLab simulations. The work presented in [91] does not consider some relevant aspects of the service migration process, such as a realistic traffic load and a realistic user mobility pattern. MobFogSim allows measuring the impact of different resource management mechanisms considering many network metrics.

Proposing a different approach for network slicing management, the authors in [161] developed a Digital Twin environment for slice management. The digital environment can accurately mirror the network behaviour and predict the E2E latency of the real infrastructure. Dynamic resource allocation and VNF migration can happen under the violation of service level agreement (SLA) requirements, *e.g.*, link failure or increasing resource utilisation. Although considering service migration and evaluating the performance of a different resource management approach, the cited work does not consider user mobility as a trigger for traffic variation, nor does it present a deep evaluation of the overhead of such operations.

The authors in [155] introduce a framework for scaling network slice's functions and resources. Due to traffic variations, the proposed framework can increase/decrease slice bandwidth, deploy/remove slice instances, or both. However, in that work, the traffic variations and the slice placement do not consider user mobility.

Two heuristics solutions were proposed for the VNF placement and scheduling problem for latency-sensitive network slices in [119]. The proposed solution determines whether to place new VNFs or to reuse the already deployed VNFs. That solution aims to optimise

Table 3.1: Scope of network slicing-related solutions.

Work	User Mobility	Service Migration	Overhead Evaluation	Objective	Contribution
[49](2022)	-	-	-	Maximise the network slice placement request acceptance ratio and minimise the total resource consumption	An algorithm for slicing placement
[114](2022)	✔	-	-	Maximise slice resource utilisation at the edge	An algorithm for slice allocation
[106](2022)	✔	-	-	Maximise long-term QoS performance of V2V services	A network slicing framework for V2V service provisioning
[29](2024)	✔	-	-	Improve Quality of Experience (QoE) for mobile users	An heuristic for dynamic bandwidth allocation
[154](2024)	✔	-	-	Predict traffic load and increase slice requests	A traffic prediction model for RAN allocation
[91](2021)	✔	✔	-	Maximise the number of migrations	An algorithm for dynamic slicing allocation
[35](2024)	✔	-	✔	Minimise system costs	A dynamic resource allocation framework for RAN slices.
[161](2022)	-	✔	-	Predict E2E latency	Digital Twin environment of network slicing
[155](2021)	-	✔	✔	Scale network slices and functions transparently to the slice end-users	A framework for dynamic network slicing
[119](2022)	-	✔	✔	Maximising profit	Two heuristics solutions was proposed for the VNF placement and scheduling
[53](2023)	-	✔	✔	Improve load balance in the core network	A SFC migration strategy
[167](2023)	-	✔	✔	Maximise profits and minimise latency	A prediction-based network slice mobility scheme
[28](2020)	-	-	✔	Minimise allocation cost in an economic perspective	A cost allocation model for network slicing
[80](2022)	-	-	✔	Minimise the deployment cost	Two heuristic algorithms for slice orchestration
[5](2022)	-	-	✔	Minimise allocation cost	A slice resource allocation framework
[63](2023)	-	-	✔	Maximise the utility of the infrastructure's resources	A comparison between different resource allocation approaches
[7](2022)	✔	✔	✔	Optimise bandwidth allocation	An heuristic for SFC migration and bandwidth allocation
[17](2023)	✔	✔	✔	Minimise latency and service migration	A service placement and migration model
[109](2024)	✔	✔	✔	Maximise profits and minimise latency	A prediction-based network slice mobility scheme
[142](2024)	✔	✔	✔	Minimising slice migration delay and maximising the slice acceptance rate	An architecture for network slice mobility management
This work	✔	✔	✔	Improve accuracy of allocation delays in resource management mechanisms	A resource allocation model and an heuristic approach for slice allocation

profits and ensure latency constraints. The overhead of VNF instantiation is considered. However, generic cost units are assumed, *i.e.*, no time or computing costs were described. Also, user mobility is out of the scope of that work.

In [53], the authors propose an aggressive migration strategy for Service Function Chaining in the core network. The proposed solution can migrate the entire SFC, aiming to improve load balance. The authors consider the migration cost in terms of processing and time in their analysis. However, no user mobility is considered as a trigger for SFC migration. Also, the cited work focuses on only one network domain, the core network.

In [167], the authors presented a prediction-based network slice mobility scheme. Based on traffic prediction, the solution suggests scaling up the slice resources or mi-

grating the slice instance, *i.e.*, service migration and network resource migration to a different region. However, despite assuming user mobility as a motivation for service migration and traffic variation, the validation of the cited work does not consider user mobility as a trigger. Also, the traffic prediction method is not further discussed in that work.

3.3 Overhead in Network Slice Resource Management

The authors in [28] present some cost allocation models for network slicing in an economic perspective. That work proposes different solutions which include a model that allocate the network cost to the different deployed slices deployed and a cost model that aims to minimise costs by using virtualisation on the core of the network. Among those results, those solutions can improve the price evaluation of different slice services. Although that work present a cost allocation model, the focus of that work is in economic perspectives. On the other hand, this work proposes an evaluation of the slice resource management in terms of the allocation delay, *i.e.*, how much time is required to perform slice operations.

The work [80] proposes a network slicing architecture and two heuristic algorithms for resource allocation and slice orchestration in manufacturing environments. That work evaluated the performance of an AI approach for the slice orchestration. A cost function of resources required to deploy VNFs is considered in the deployment evaluation. The deployment cost is composed of node cost and link cost, in which the node cost is composed of the cost of computing resource deployment for all nodes while the link cost is composed of bandwidth allocation cost for all the slice links. Similar to our work, the proposed work considers the overhead cost of network slice operations. However, the cost function does not describe the required time to perform such operations. Furthermore, dynamic traffic variation triggered by user mobility is not considered to evaluate the performance of the proposed approach in a dynamic scenario.

The work presented in [5] proposes one a resource allocation framework for end-to-end dynamic network slicing. The work focuses on latency, availability, and reliability requirements. The distributed solution manages inter-slice and intra-slice allocations. In addition, the placement of VNF is also considered. A monetary cost is considered for the allocation process. Although considering the evaluation of dynamic end-to-end network slicing, the work does not include the feasibility of the proposal in a higher dynamic scenario, such as the one seen for mobile users, nor considers the operational cost, such as computing and time, in the allocation process.

The work in [63] presents an comparison between different resource allocation approaches for end-to-end network slices. Different deep reinforcement learning (DRL) algorithms are considered. The authors aim to maximise the utility of the infrastructure' resources. Although that work considers a traffic variation in the resource allocation model, both the number of users requests and the data rate requests are defined randomly. In our work, the traffic variation is defined by the user mobility, which is, in particular, based on real user mobility patterns. In addition, that work does not consider the impact of delay allocation in the network performance.

The authors in [7] proposed a Deep Reinforcement Learning solution for migration of VNFs migrations. Based on user mobility, VNFs can be migrated to improve bandwidth allocation. A cost evaluation in terms of downtime is considered. A testbed infrastructure is used for validation. However, the mobility of the users is limited in that approach.

The work presented in [17] proposes one solution for resource management for mobile users in edge-cloud infrastructures. Similar to our work, the authors propose VNFs/Service migrations to improve service placement and minimise the service latency. A second step aims to minimise the service migration as much as possible to avoid service outages and computing overhead. Finally, the main objective is to suggest a balance between those two objectives. An overhead analysis is considered in terms of migration time and computing processing. Although considering user mobility as motivation, the validation of the cited work assumes a random walk process as a user mobility pattern. Also, a bandwidth slice reallocation is out of the scope of that work.

An extension of [167] is presented in [109]. The authors provide a further study regarding the traffic prediction method. A DRL solution that learns users' requests and position information is proposed. Those data are used in the slice mobility process. Differently from their previous work, in [109], the authors consider user mobility in the validation. However, the authors assumed a random mobility pattern to evaluate the proposed approach.

The authors of [142] propose a cloud-native microservices architecture for network slice mobility management in edge computing environments. Due to user mobility, services and slice instances might need to be migrated to different edge servers to ensure service continuity and QoS requirements. One slice migration delay is considered in the migration process. In that work, that overhead "is the time taken to complete the slice mobility, *i.e.*, the microservices required for a service-specific dedicated slice, in-memory state, and configuration files are transferred to the candidate MEC server". Despite considering a service migration scenario for user mobility support in edge environments, dynamic resource allocation is out of the scope of that work.

As introduced in this chapter, network slicing has received attention in resource allocation topics in recent works. However, despite some of those solutions considering the impact of reallocation delay in its algorithms, an in-depth assessment of the delay impacts in realistic user mobility scenarios is still missing. This work is the first to evaluate the impact of different delays in the slicing allocation problem in a scenario of traffic variations triggered by realistic user mobility. Furthermore, this work is, for the best of our knowledge, the first to propose a resource allocation model to describe the required time to proceed with the slicing allocation process.

3.4 Network Simulators

Network simulators have been present as a suitable environment to validate resource management mechanisms in large-scale scenarios. In this context, some solutions have been proposed in recent works. However, each simulator has strengths and weaknesses aspects in the resource management of fog computing infrastructures. This section aims

to introduce the main network simulators related to fog computing environments available in state of the art and present how MobFogSim is compared to them.

Recent surveys, from 2020 to 2024 [101, 100, 102, 64, 86, 34, 51, 176], present a further literature review about simulation environments for 5G/edge/cloud computing research. All these works highlight the need for simulators with mobility support and the value of supporting the evaluation of resource management mechanisms. The set of simulators cited in this section was based on those surveys. Table 3.2 summarises the main characteristics of those simulators. The table highlights the simulators' characteristics based on their support for device mobility, device-to-device communication, service migration, and network slicing.

Simulator	**Device mobility**	**D2D comm**	**Service migration**	**Network slicing**	**Extends**
iFogSim [71]	-	-	-	-	CloudSim
YAFS [90]	✔	-	-	-	-
VirtFogSim [139]	✔	-	-	-	MATLAB
EasiEI [152]	✔	-	-	-	NS-3
EdgeCloudSim [149]	✔	✔	-	-	CloudSim
FogNetSim++ [125]	✔	✔	-	-	OMNeT++
PureEdgeSim [105]	✔	✔	-	-	CloudSim+
IoTSim-Edge [79]	✔	-	✔	-	CloudSim
DISSECT-CF-Fog [103]	✔	-	✔	-	DISSECT-CF
EdgeSimPy [150]	✔	-	✔	-	-
EdgeAISim [112]	✔	-	✔	-	EdgeSimPy
iFogSim2 [95]	✔	✔	✔	-	iFogSim
EdgeSim++ [93]	✔	✔	✔	-	NS-3
Simu5G [113]	✔	✔	✔	-	OMNeT++
Py5cheSim [118]	-	-	-	✔	-
SliceNet [87]	-	-	-	✔	-
SliceSim [4]	✔	-	-	✔	-
MobFogSim	✔	✔	✔	✔	iFogSim

Table 3.2: Comparison of network simulators related to the context of this work.

iFogSim [71] was the first fog computing simulator. It is implemented in Java as an extension of the most popular cloud computing simulator, CloudSim[1]. iFogSim allows to test resource management and service placement strategies in terms of (i) service latency, (ii) service throughput, (iii) network usage, (iv) energy consumption, (v) operational costs, and (vi) pricing. iFogSim provides a GUI to describe fog network topologies (*i.e.*, sensors, actuators, fog nodes, cloud data centres, and interconnections among them). Topologies can then be exported as a JSON file and imported again later.

iFogSim presents several strengths, which are the reasons why it was chosen to implement MobFogSim as its extension. Firstly, iFogSim provides the broadest range of functionalities, which span from network and resource management modelling to energy

[1] `https://github.com/Cloudslab/cloudsim`. Last accessed: April 10th, 2024.

consumption and operational costs modelling. Secondly, it is by far the most used fog simulator in literature, making it the best candidate to compare its solutions with the related work. Thirdly, iFogSim extends CloudSim, which is the most popular cloud computing simulator. Therefore, it is relatively easy to use for all those who already have experience with CloudSim. However, iFogSim also has some limitations. The most evident is the lack of detailed and consistent documentation, which might make iFogSim hard to use for those who have never worked with CloudSim. Besides, network modelling in iFogSim is rather simplistic, as this simulator does not deal with real-network aspects such as packet loss, network congestion, and channel collisions. Furthermore, iFogSim does not model mobility scenarios and VM/container migration among fog nodes. To overcome some of the above limitations, researchers have recently developed iFogSim2 [95] as an extension of iFogSim. This new simulator version provides further functionalities such as device mobility, VANETs modelling, and fog service migration but still needs to model network slicing.

Yet Another Fog Simulator (YAFS) [90] is a Python simulator for fog computing environments. It is particularly good at modelling network failures and evaluating service placement solutions in failure cases or designing robust networks. Network failures may be modelled in two possible ways: (i) through the runtime creation/deletion of fog nodes and network links; (ii) through custom processes, namely functions invoked at runtime for the implementation of real events. YAFS models mobility, sensors, and actuators but does not include aspects such as energy consumption or VM/container migration. Finally, although it does not include a GUI to describe fog network topology, YAFS allows importing a simulation scenario as a JSON file.

VirtFogSim [139] does not model aspects such as VM/container migration, energy consumption, or pricing. However, it dynamically tracks the energy-delay application performance against abrupt changes due to failures or device mobility, *e.g.*, mobility-induced changes of the available up/down bandwidth. The most specific functionality of VirtFogSim is that it allows modelling cellular network access, which is useful when simulating 4G/5G scenarios. VirtFogSim is currently the only simulator that explicitly provides such a feature. Besides, this simulator includes a Graphical User Interface (GUI) that shows the simulation results in tabular, bar-chart, and coloured map graph formats.

Differently from most of the edge computing simulators presented in the literature, EasiEI [152] is a simulator not based on CloudSim. EasiEI is based on NS-3 instead [130]. The simulator is written in C++ and incorporates the network management from its core network, as well as energy consumption and user mobility features. Focusing on task scheduling, network resource management, and monitoring, EasiEI focuses on complex IoT scenarios. Although presenting some unique features in the literature, the simulator still lacks some features, such as service migration and network virtualisation, like network slicing.

EdgeCloudSim [149] is another prominent simulator for fog computing environments. Also, EdgeCloudSim is an extension of CloudSim. EdgeCloudSim models network delays more accurately than iFogSim, which considers network delays to be permanently fixed. EdgeCloudSim, instead, includes a networking module that calculates network delays based on the current network load when data needs to be sent over the network.

Even though EdgeCloudSim provides better network modelling, it does not provide the same range of functionalities available in iFogSim. For example, energy consumption, operational costs, and pricing models are all missing in EdgeCloudSim. Device mobility is modelled, but VM/container migration is not. Hence, EdgeCloudSim may be helpful to those working in Java and primarily focused on evaluating network metrics (*e.g.*, service latency and network usage). Device mobility is modelled; however, VM/container migration and dynamic network slicing are not.

The main objective of FogNetSim++ [125] is to overcome the limitations of the other simulators in network modelling. They do not (or only partially) consider real-network properties and, therefore, simulate idealistic networks without packet loss, congestion, or channel collision. Instead, FogNetSim++ extends OMNeT++[2], a well-known framework for building network simulators to model all these aspects. Moreover, it includes popular communication protocols for simulation, such as TCP, UDP, MQTT, and CoAP. Furthermore, FogNetSim++ models several other aspects, such as energy consumption, pricing, mobility, and handoff mechanisms. However, it does not model VM/container migration and dynamic network slicing.

PureEdgeSim [105] is simulator based on CloudSim Plus [146]. The key aspect of the simulator is enabling device communication and sharing resources. PureEdgeSim integrates heterogeneous IoT devices and Edge and Cloud resources, supporting service offloading and device mobility. Although offering Device-to-Device (D2D) and device mobility similar to MobFogSim, PureEdgeSim does not support service migration.

IoTSim-Edge [79] is another simulator extending CloudSim. IoTSim-Edge supports modelling different Internet of Things (IoT) protocols and their energy consumption profile. Besides, it models device mobility and fog service migration. However, it does not model VANETs and network slicing.

Another relevant fog computing simulator is DISSECT-CF-Fog [103] [3], which is based on DISSECT-CF [85], which models an environment to cloud, fog, IoT devices, sensors, and its communications. The simulator models its environment as Infrastructure-as-a-Service, providing several different states to its physical resources, virtual machines and network communications. The network is modelled in terms of latency, and input and output bandwidth between the nodes. DISSECT-CF-Fog does not support user mobility. Besides, the simulator does not provide any network slicing mechanism.

EdgeSimPy [150] is a simulator written in Python focused on the life cycle of container-based applications on edge computing infrastructures. Like MobFogSim, EdgeSimPy supports device mobility, service placement, and migration features. That simulator supports some relevant aspects of network simulation, such as energy consumption and routing. However, despite implementing a virtualised environment as a container, that tool does not implement network virtualisation in the form of network slicing.

Extending EdgeSimPy, EdgeAISim [112] offer a toolkit for resource management in Edge focusing on AI-based solutions. EdgeAISim supports energy management, service migration, and mobility support like its core simulator. Based on AI solutions, the simulator improved energy consumption in task migration scenarios. However, network slicing

[2]See `https://omnetpp.org/`. Last accessed: April 10th, 2024.

[3]https://github.com/sed-inf-u-szeged/DISSECT-CF-Fog. Last accessed: April 10th, 2024.

support is missing.

Similarly to EasiEI, EdgeSim++ [93] is a simulator based on NS-3. Both simulators are focused on task scheduling and network management. EdgeSim++ manages an IoT-Edge-Cloud infrastructure and supports user mobility and service migration. That simulator also provides additional features, such as device-to-device interaction and support for AI algorithms in the resource management process. The simulator also offers a tool for result visualisation. Although offering a simulation environment with support for several network scenarios and validation mechanisms, EdgeSim++ does not support network slicing.

Simu5G [113] is a library for the OMNeT++ simulation framework that provides a collection of models to build arbitrarily complex simulation scenarios. Specifically, Simu5G models the data plane of the 5G Radio Access Network (RAN) and core network. Besides, it models device mobility, VANETs, fog service migration, and the Cellular-Vehicle-to-Everything (C-V2X) standard. However, Simu5G does not model network slicing scenarios.

Focusing on simulating 5G cells, Py5cheSim [118, 117] is one of the few simulators that models network slicing at the Radio Access Network (RAN). One of the core features of the simulator is that it allows inter- and intra-slice scheduling mechanisms. Py5cheSim also supports a set of AI algorithms. Although Py5cheSim is one of the few simulators that supports network slicing in the literature, that tool does not support user mobility, which is a key aspect in 5G/network slicing scenarios.

SliceNet [87] is a network slicing simulator focusing on resource management. The simulator network models edge and cloud environments, network functions, and the allocation of multiple slices with different characteristics in parallel over the same "physical infrastructure". The simulator also presents a tool for visualisation of ongoing simulations and simulation results. Like MobFogSim, SliceNet models the slices as a percentage of available resources. Also, both simulators support static and dynamic network slice allocation in an end-to-end slice. However, SliceNet does not support device mobility, a key aspect in network slicing scenarios.

Finally, SliceSim [4] is a discrete-event simulator written in Python that models device mobility and network slicing, even though only for the RAN part of the network. Nonetheless, SliceSim does not model VANETs, fog service offloading or fog service migration.

MobFogSim is one of few simulators to model VM/container migration, which is important for supporting device mobility and other reasons such as load balancing. In addition, there is not any simulator that supports network slicing in that scenario. An extension [66] of MobFogSim implements dynamic network slicing, which is not provided by any other fog computing simulator. The solution developed in this thesis, *MobFogSim* [121], fills this gap in the literature. MobFogSim thus inherits all the functionalities from iFogSim, extending this simulator with mobility modelling, VM/container migration, and dynamic network slicing.

In light of the above discussion of this section, there are several network simulators for edge-cloud infrastructures. In this context, among the 20 simulators considered in this section, four characteristics were highlighted and discussed in this work, as seen

in Table 3.2: 1) Device mobility support, which is a common feature in 15 from 18 simulators; 2) device-to-device communication, which is offered in 7 simulators; 3) service migration, that is supported in 8 simulators; and 4) network slicing support, which seems in 4 simulators from that universe of 20 tool evaluated in this work.

As presented in this section, besides the existence of some simulators with mobility support, service migration support, device-to-device communication and network slicing support, MobFogSim is the only one that can combine these characteristics.

In the next chapter of this thesis, it is presented further details of the MobFogSim simulator. The development of that simulator is one step to building a reliable simulation environment to validate the proposed solutions presented in this thesis. In this scenario, MobFogSim allows us to evaluate the impact of network slicing as a solution for user mobility support in fog computing environments. Chapter 4 presents in detail the core architecture of MobFogSim, its features, such as user mobility support, service migration, and network slicing, and its validation in terms of calibration and scalability.

Chapter 4

MobFogSim - Simulation of Mobility and Migration for Fog Computing

One fundamental step in the development of this thesis is the use of one reliable environment to validate the solutions for resource management which were developed in this work. Facing economic limitations in adopting the testbed approach, one simulator, named MobFogSim, has been developed for that purpose. Previously in this document, Section 3.4 presented a discussion regarding MobFogSim contributions in the literature. In addition, this chapter presents further details regarding that simulator, describing its architecture and features. The experiments based on a testbed used for validating the simulator are also presented in this chapter.

MobFogSim [121], derived from MyiFogSim [92], is an open-source fog computing simulator[1] developed as an extension of iFogSim [71]. The latter is written in Java and is widely used for modelling and simulating fog computing networks. Initially, MobFogSim was focused on modelling edge-cloud infrastructures with user mobility support. In this thesis, that release is referred as *MobFogSim 1.0*. Applications in MobFogSim are modelled as Directed Acyclic Graphs (DAGs) where vertices represent application modules, while edges represent interactions expressed as offloading of tasks (called "tuples" in iFogSim) between modules. In terms of "physical infrastructure", iFogSim models IoT devices, access points, fog nodes, and cloud data centres. MobFogSim model generalised aspects related to device mobility and VM/container migration in the fog, *e.g.*, user position and speed, connection handoff, migration policies, and strategies, to name a few. Those features compose the MobFogSim 1.0. Further details are present in the sections 4.1, 4.2, and 4.3. In a continuous development, new features have been included in the simulator. As a result, MobFogSim is the only available simulation environment that supports user mobility, service migration, network slicing, and vehicular networks. Those features compose the current simulator release and it is referred in this thesis as *MobFogSim 2.0*. Further details regarding those features are present in the sections 4.4, 4.5, and 4.6. A discussion about the contributions of MobFogSim in the face of the state-of-art simulators was present in Section 3.4. An overview of MobFogSim is present in the remainder of this section.

[1]See `https://github.com/diogomg/MobFogSim`. Last accessed: April 10th, 2024.

For the sake of clarity and convenience, the content present in this chapter results in contributions for the following articles: [121], [66], [68], and [70]. Those cited articles may present a different presentation of MobFogSim's architecture, procedures, experiments, and validation scenarios.

The remaining of this section is organised as follows:

- Section 4.1 presents the basic architecture of MobFogSim;
- Section 4.2 presents an overview of the service migration process on MobFogSim;
- Section 4.3 presents the user's mobility support of MobFogSim;
- Section 4.4 describes how MobFogSim was extended to include dynamic slicing support;
- Section 4.5 introduces the end-to-end slicing and vehicular network support;
- Section 4.6 presents the MobFogSim performance regarding resource requirements and scalability;
- Section 4.7 presents some preliminary experiments using a testbed to validate the simulator;
- Section 4.8 summarises this chapter and presents the conclusions and final remarks.

4.1 Basic Architecture

Compared with iFogSim, MobFogSim adds some classes to its architecture. These classes make MobFogSim an extension of iFogSim with support for device mobility and VM/container migration. Figure 4.1 illustrates the main classes in MobFogSim and highlights those not present in iFogSim. In that figure, the classes are illustrated as a class diagram in the Unified Modelling Language (UML). The leftmost part represents classes from the CloudSim simulator, which is important for creating the relevant entities in iFogSim and MobFogSim. Classes from the CloudSim simulator are responsible for creating and managing cloud entities, such as servers and virtual machines and their characteristics. Also, the CloudSim package includes cloud operations, such as monitoring servers' load. Detailed comments regarding CloudSim classes are described below.

- *SimEntity:* This class implements the basic structure to describe entities in the simulation. Each *SimEntity* object has attributes, such as id, state, and the set of events (actions) to be executed. A *SimEntity* object can manage its own events and send events to other entities;
- *DataCenter:* In the simulator, a data centre is modelled as a set of virtualised resources. A *Datacenter* class provides a structure to manage that set of virtual machines, describing the static characteristics of its resources, such as operating system, management policy, resources' cost and the time zone the resource is located;

- *PowerDatacenter:* This class structures runtime operations in the server, such as energy consumption, migration process and load capacities.

In addition to Figure 4.1, the classes from iFogSim reside in the middle of the picture. IFogSim package includes edge device entities, such as edge servers, access points, sensors and actuators. Furthermore, the iFogSim package includes resource management operations, such as VM placement and migration.

- *Sensor:* This class implements the source of inputs linked to an application running on a fog device, *e.g.*, a keyboard or a temperature sensor. In the simulator, a sensor describes the latency to collect and send data;

- *Actuator:* This class implements the output of an application running on a fog device, *e.g.*, a smartphone screen or a traffic light;

- *FogDevice:* This class extends *PowerDatacerter*. In iFogSim, both edge devices (servers) and user devices (IoT devices) use *FogDevice* class to be implemented. A *FogDevice* has a set of virtual machines, which one with its characteristics, such as resource capacities and current load. In the simulator, the main applications run on fog devices located at the edge of the network (server); meanwhile, IoT devices run the user-side version that interacts with the server side.

Finally, in Figure 4.1, the rightmost part shows the main classes introduced with MobFogSim. These classes allow modelling device mobility, VM/container, and the network handoff.

The description of MobFogSim classes is present as follows:

- *Coordinate:* This class acts as a Cartesian Plan map (X, Y) to have all entities position during the simulation. These entities can be the fog servers, the wireless base stations (Access Points - APs), and the users' (IoT) devices. The developer can configure the map boundaries;

- *ApDevice:* This class extends *FogDevice* and has the access point responsibility in a wireless network. This class manages the handoff mechanisms and connections/disconnections of end devices;

- *MobileDevice:* This class also extends *FogDevice*. Its main goal is to allow a separation between fog servers and IoT devices since iFogSim implements any device (servers and IoT) with the same features. With this separation, it is possible to have specific features for different devices;

- *MobileSensor:* This class extends the *Sensor* class. In iFogSim, the developer needs to instantiate several *Sensor* objects to build a scenario with multiple sensors. In MobFogSim, *MobileSensor* already has a set of sensors, and the developer can instantiate only one object and add, if necessary, more sensors into the same hardware. A *MobileSensor* is associated with a *MobileDevice*;

- *MobileActuator:* This class is at the same level of abstraction as *MobileSensor* and extends the Actuator class from iFogSim;
- *MigrationStrategy:* This class implements the migration strategy to be applied in the migration process. The migration strategy defines the metric to find the destination node in the migration process. Further details are presented below;
- *MigrationPolicy:* This class implements the migration policy to be applied in the simulation. In the simulation, the migration policy defines when a migration starts. Further details are presented below.

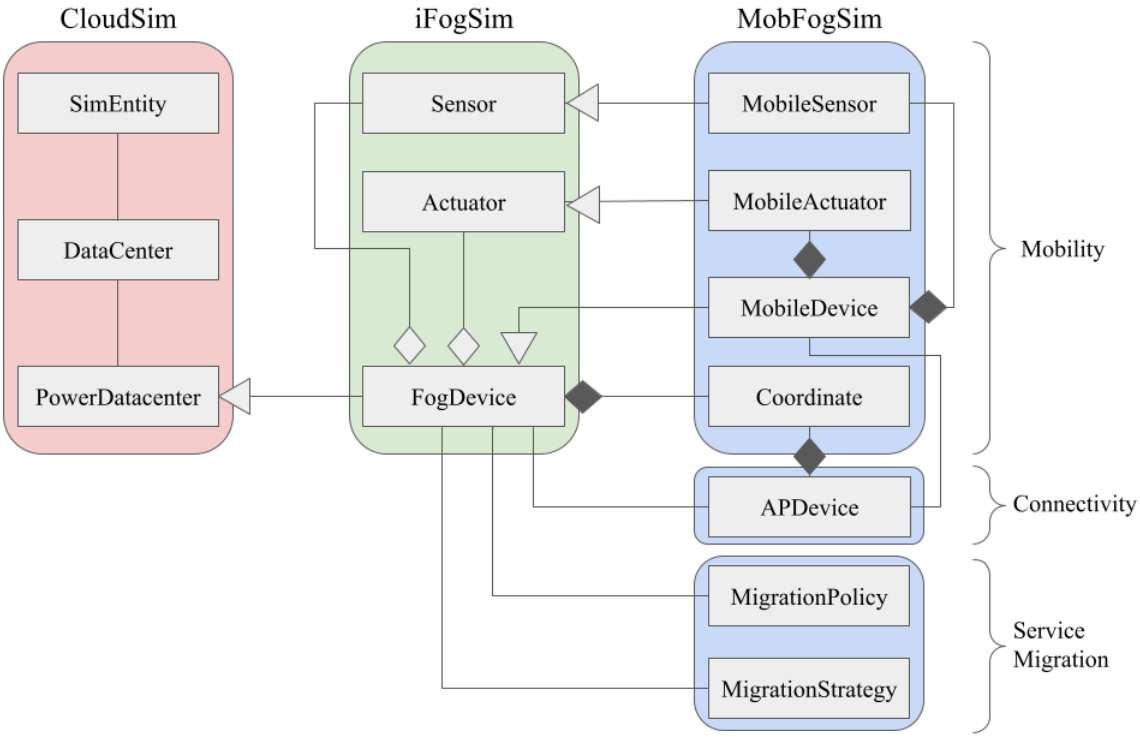


Figure 4.1: Overview of MobFogSim as an extension of iFogSim and CloudSim.

These classes, illustrated in Figure 4.1, implement the migration model, thus creating a mobile simulation environment in MobFogSim that includes the decision-making algorithms, the events related to the migration, and the localisation system. The migration algorithm, especially the migration decision-making, needs input from a set of parameters to implement a migration policy and a migration strategy, as detailed below.

- Migration strategy: A migration strategy is a placement algorithm that decides which is the next fog node to receive a VM/container that will be migrated. That strategy can be modelled in several ways, using different optimisation techniques and input data from the system and the users/services. Currently, MobFogSim implements three different migration strategies based on network conditions and user location-aware metrics. These migration strategies are provided as examples since

developers can implement their own algorithms, which can consider specific and, more or less flexible, QoS levels. The migration strategies currently implemented in MobFogSim are:

1. The lowest distance between the user and the access point - which chooses the fog node connected to the access point that is closest to the user;
2. The lowest distance between the user and the new fog node - which chooses the fog node that is closest to the user;
3. The lowest latency - which chooses the fog node with the lowest latency to the user.

- Migration policy: A migration policy can consider the geographical context of a user and her/his device in order to trigger a migration:
  - User's position on the map
  - User's speed and direction
  - Migration zone: an area related to AP's coverage area where the migration decision is computed.
  - Migration point: a location in the map where the computed migration can be performed.

The Migration Policy has a relevant role in the Migration Process. MobFogSim defines an area (Migration zone) where migration decisions are constantly computed. Once a migration is defined, the migration starts when the user leaves the Migration Zone, crossing the Migration Point. The Migration Point can be defined at a constant distance within the AP' boundary or be dynamically defined based on user speed and VM size. Figure 4.2 illustrates the access point modelled in MobFogSim and its relationship with Migration Policy. Further details about the migration process implemented on MobFogSim are present in Section 4.2.

4.2 Migration Process

Based on the Migration Strategy and Migration Policy, MobFogSim defines the triggers for starting a migration. The migration process uses the network resources in the form of slices (fog nodes and links), the user's devices (mobileDevice), and the chosen migration strategy as input. The relationship between service migration and network slices in MobFogSim is described in Section 4.4. Algorithm 1 presents the necessary steps to perform the migration process on MobFogSim.

In the migration process, while the user is within the migration zone (Figure 4.2), the infrastructure computes the migration decision to define, if there is, the destination fog node to receive the user's service. The Migration Decision process (line 2 in Algorithm 1) is the one that makes the migration decision based on policies and strategies established by the migration system. Migrations occur when the system identifies a better fog node to place the user's service based on the metric defined in migrationStrategy, *e.g.*, lowest

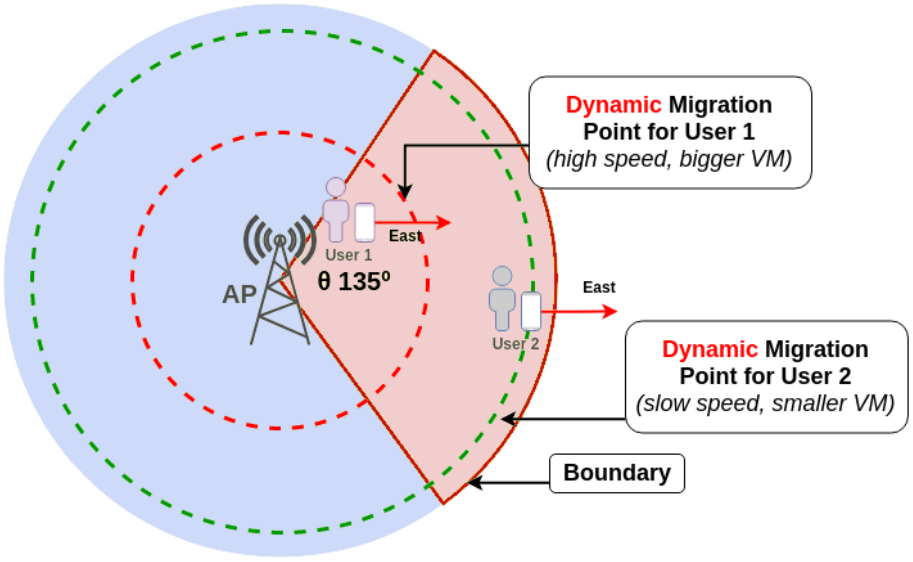


Figure 4.2: Visual concept of access point area and its relationship with MobFogSim's Migration Policy.

Algorithm 1: Migration Process in MobFogSim.

```
   input : slices, mobileDevice, migrationStrategy
1  while mobileDevice is within the migration zone do
2      mobileDevice.destinationFogNode = migrationDecision(slices.fogNodes,
         mobileDevice, migrationStrategy);
3      if mobileDevice.destinationFogNode != mobileDevice.currentFogNode then
4          if mobileDevice crosses the MigrationPoint then
5              startMigration(slices, mobileDevice.sourceFogNode,
                 mobileDevice.destionationFogNode);
6              while migration is being performed do
7                  requestSliceReshaping(slices, mobileDevice.sourceFogNode,
                     mobileDevice.destionationFogNode);
```

latency, lowest distance, or another metric. In scenarios where no fog nodes offer better conditions to the user, the current fog node remains the only alternative to run the user's service. If the policy/strategy judges that it is necessary to perform a migration process (a true return from the condition in line 3), the migration point, which defines how close the user is to the wireless network boundary, defines when the migration starts. When the user crosses the migration point (line 4), the system can start the migration. Once these conditions (where and when) are satisfied, the system starts the migration (line 5). If anything impairs VM/container migration, either the management system tries to solve the problem at runtime, or the migration might be aborted, and the migration process finishes, leaving the VM/container on the source fog node. On the other hand, if everything runs as expected, the migration process goes to the After Migration phase to liaise with the user's mobile device to close the connection with the old fog node and use the new fog node. If considering a scenario with dynamic slicing while the

migration is being performed (line 6), the infrastructure can reallocate more resources for the slice that is serving that service migration (line 7). If that request received a positive answer, the mobileDevice's slice would access more bandwidth to perform that migration. Otherwise, the migration will occur using the currently available bandwidth. Further details about network slicing support on MobFogSim are present in Section 4.4, *i.e.*, requestSliceReshaping() process in line 7 of Algorithm 1.

4.3 Realistic User's Mobility Support

The modifications made to iFogSim, which resulted in MobFogSim, introduced support to mobile devices. Aiming to enrich the mobility scenarios, these modifications enable simulations to be carried out using data from realistic mobility databases. An example is the Luxembourg SUMO Traffic (LuST) database [31], which contains data from vehicle mobility in Luxembourg. In that context, a mobility simulator can interpret these databases. The workflow is as follows.

Figure 4.3 presents an example of the data flow used in a simulation. In that example, the mobility tool *Simulation for Urban Mobility* (SUMO) [20] (b) interprets the source mobility database saved in, for example, .XML format (a). Many realistic mobility databases are available in .XML format, such as the project LuST [31], which presents real data from vehicles in Luxembourg. The result of SUMO is then saved in .csv format (c).

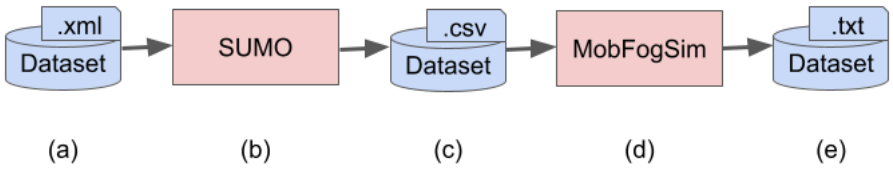


Figure 4.3: Example of the data flow in a simulation.

Each .csv file represents the mobility of a vehicle in the simulation. Each line in this file contains data from the vehicle at one point in the simulation. The data are (i) position x and y on the map, (ii) speed in meters per second, (iii) the direction in radians, and (iv) the simulation time in which these data were collected.

Hence, this new database is used to define the users' mobility in MobFogSim. The simulator interprets this database and makes some modifications to adapt these data to its mobility model. Among these changes, there are, for example, the conversion of the vehicle speed from meters per second to kilometres per hour and the conversion of the direction from radians to the eight main cardinal points, which are the basis of the mobility model in MobFogSim.

After the simulation in MobFogSim (d), a new database is built (e), presenting the user's behaviour results for local resource management. Among these results, there are (i) the average and the maximum latency presented by the services along the user's path, (ii) the migrations performed, (iii) the packets requested and attended, and (iv) the number of handovers.

4.4 Dynamic Network Slicing Support

Based on MobFogSim's basic architecture, an extension was developed to introduce dynamic network slicing support on that simulator. The contribution of that extension and the set of experiments presented as a validation scenario were published in [66]. Basic concepts regarding resource allocation in dynamic network slicing are presented in Section 2.2.2. Implementation details of the MobFogSim release are discussed in this section. In addition, the validation scenarios regarding dynamic network slicing are further discussed in Chapter 5.

In MobFogSim, infrastructure resources can be categorised into network and cloud/fog resources. The latter refers to storage and computing resources from cloud servers and fog nodes and are provided as VMs or containers. Network resources in MobFogSim are currently expressed in terms of the bandwidth available in the links. These links are from access and transport networks. The transport network consists of the interconnections among fog nodes and access points, while the access network is related to the wireless interfaces that connect users to the fixed infrastructure.

Figure 4.4 illustrates the overall MobFogSim architecture, highlighting the relationship with the classes inherited from iFogSim and CloudSim. The illustration of MobFogSim modules presented in Figure 4.4 complies with a class diagram in the Unified Modelling Language (UML). The empty diamond represents the aggregation relationship, *e.g.*, *Sensors* and *Actuators* are part of *FogDevices*. The filled diamond represents the composition relationship, *e.g.*, the *FogDevice* and *APDevice* have one *Coordinate*. There is no *FogDevice* that does not have a *Coordinate* defining its place in the map. The triangle symbol represents the Inheritance relationship, *e.g.*, a *FogDevice* is a kind of *PowerDataCenter*. Modules highlighted in grey in Figure 4.4 were modified to support static and dynamic network slicing and vehicular network support, as discussed in this section and Section 4.5.

Network slicing in MobFogSim is implemented by orchestrating resources among slices. All the slices thus share the same physical infrastructure in the simulator; however, each slice reserves a pool of resources. Resources reserved by a slice are dedicated to users within that slice. MobFogSim achieves slicing support by extending two elements in the simulator's architecture. The first element is a new Java class called *Slice*, which is located in the *org.fog.placement* package. That class defines the portion of network bandwidth granted to each slice. Values are currently expressed as a percentage of the overall bandwidth. However, modifications can be made in the simulator to use absolute values. Moreover, *Slice* specifies the groups of users that have access to its slice resources. Multiple groups can indeed be assigned to the same slice. The second element introduced in MobFogSim is a new attribute, called *groupid*, associated with the *MobileDevice* class. This attribute is used as a tag to identify the group of users to which that specific mobile device belongs. These groups are just symbolic and represent the different users' characteristics and requirements defined in the simulation setup. Allocation of VMs/containers in this extended version of MobFogSim follows the same rules as in the previous release [121]. Each user is provided with a VM/container described in terms of processing and storage attributes. It is possible to grant different cloud/fog resources based on *groupids*. Thus, the *Slice* class in MobFogSim describes the list of fog nodes and their respective re-

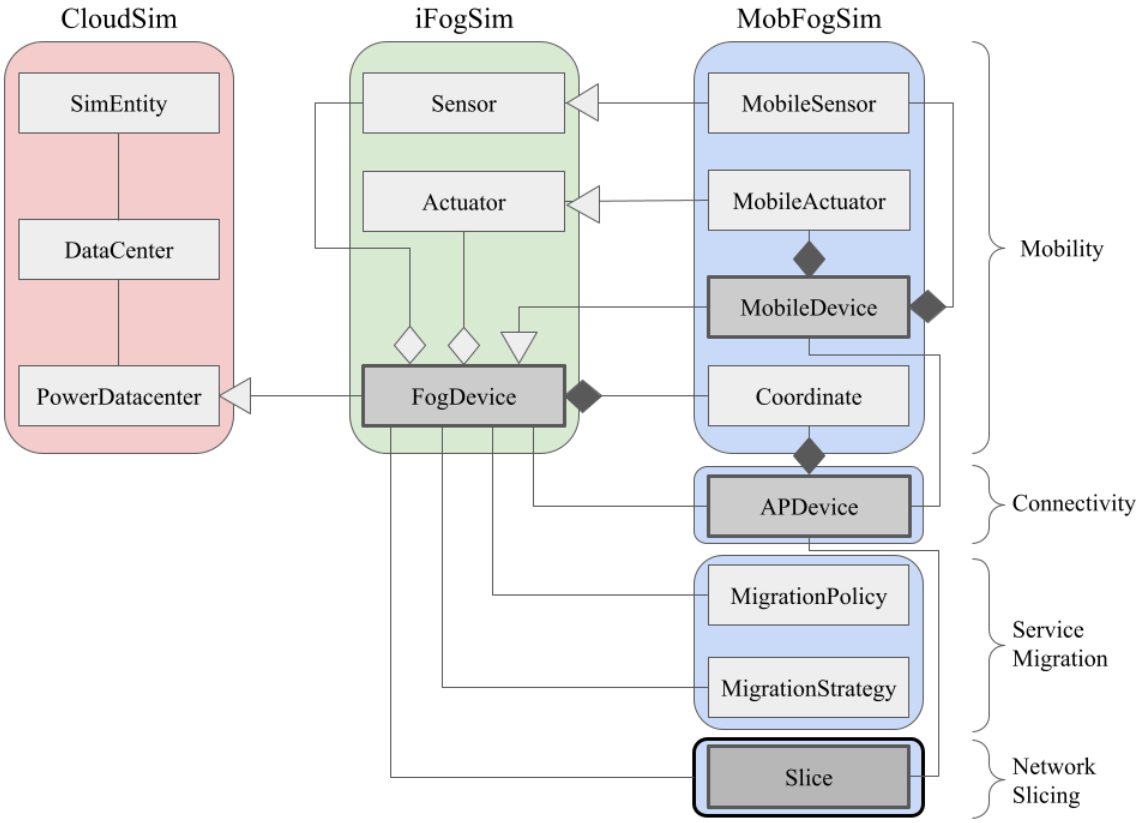


Figure 4.4: Logical architecture of MobFogSim, which extends Cloudsim and iFogSim.

sources that one slice may access. However, an analysis of the impact of different amounts of cloud/fog resources for each slice is out of the scope of this work.

In MobFogSim, the overall network bandwidth is split among slices in terms of packets. Namely, based on the portion of bandwidth assigned to a slice (as specified by the *Slice* class), the simulator selects that percentage of traffic for that slice. For instance, if a slice instance claims 30% of the whole bandwidth, 30% of all packets transmitted on the transport network are reserved for users of that slice.

The extension of MobFogSim also supports dynamic network slicing, *i.e.*, dynamically changing the resource distribution among slices. In this thesis, the *dynamic slicing allocation problem* is investigated, focusing on a support traffic variation triggered by user mobility scenarios. Different factors can result in an increasing traffic variation. However, this work assumes a service migration between two different physical nodes as a significant event in the evaluation scenario. In the assumed scenario, the network infrastructure adopts the follow-me cloud concept. Then, some services may be migrated from one source fog node to another to serve its mobile users as closely as possible. In this scenario, a network slice can request a resource reallocation to scale-up its link bandwidth to proceed with the migration process. This process is present in the Algorithm 1 and was further discussed in Section 4.2.

In this scenario, once the NFV Management and Orchestration (MANO) system detects an increasing traffic variation, it can start the slice reallocation process. Figure 4.5 illustrates a flowchart of the slice reconfiguration process in a traffic variation scenario assumed in this work. It is assumed that the first attempt to serve the slice is allocating

resources not assigned to any other slice. If there are no available resources in the physical node, the MANO system can identify idle resources allocated in other slices and start the process of resource redistribution. In this case of underutilisation of network resources, idle resources are deallocated from other slices and assigned to that new one.

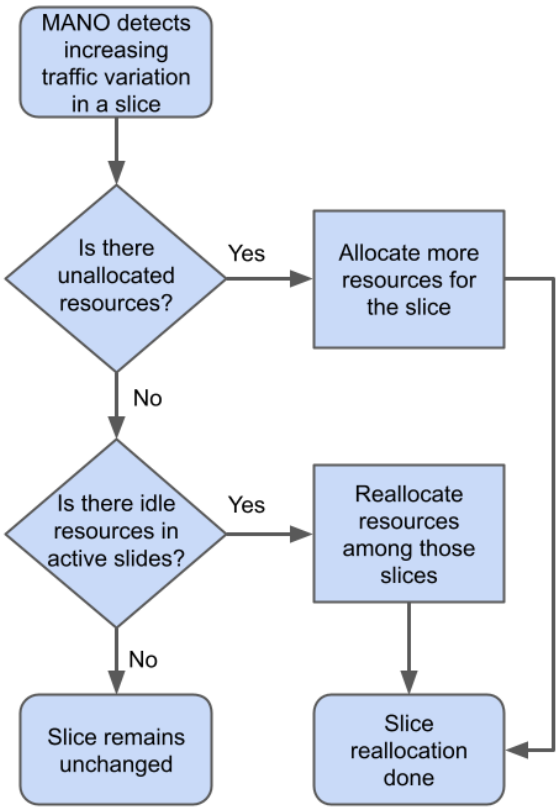


Figure 4.5: Flowchart of slice reconfiguration process due to increasing demand.

It is important to highlight that only bandwidth resources can be dynamically reallocated among slices in the current release of the simulator. Hence, idle bandwidth resources may be reassigned at runtime to other slices that need more bandwidth to attend to their users' requirements. It is worth noting that resource reallocation is not immediate, as it always incurs a reallocation time due to computing overhead. In MobFogSim, that time is modelled as a parameter rather than a constant since each simulation setup may experience a different reallocation time. Specifically, the value of this parameter can be either inserted by MobFogSim users or calculated by a function in the simulator. As a result, static network slicing does not allow the reallocation of resources; however, such resources are immediately available to slices once they request them. On the other hand, dynamic network slicing enables the reallocation of resources at runtime, but this reallocation experiences a delay. A further evaluation regarding the overhead present in the dynamic slicing operations is presented in Chapter 6.

The necessary steps to perform the slice reallocation process triggered by a migration process on MobFogSim are presented in Algorithm 2. Figure 4.5 illustrates those steps of Algorithm 2.

In MobFogSim, the migration process uses as input the network resources in the

Algorithm 2: Network slice bandwidth reallocation in MobFogSim

```
   input : slice, setOfSlices, sourceFogNode, destinationFogNode
1  if isThereAnIncresingTraffic Variation() then
2      link = linkPresentingAnIncresingTrafficVariation();
3      if isThereUnallocatedBandwidth(slice, link) then
4          allocateBandwidthForSlice(slice, link);
5      else
6          if isThereBandwidthUnderutilisation(slices, link) then
7              deallocateBandwidthFromSlices(slice, setOfSlices, link);
8              allocateBandwidthForSlice(slices, link);
```

form of slices (fog nodes and links), the user's devices (mobileDevice), and the chosen migration strategy (shortest distance from edge node, shortest distance from access points, or lowest latency). Algorithm 2 is responsible to monitor the network traffic, identifies increasing demands on slices, and performs slice reconfigurations in order to improve network performance. Figure 4.5 illustrates those steps. Once the infrastructure identifies the beginning of a migration process, *i.e.*, an increasing traffic demand in the slice's link (line 1 in Algorithm 2), the process of increasing the available bandwidth starts. Based on the link with an increasing traffic demand identified (line 2 in Algorithm 2), the MANO system tries to increase bandwidth availability to that slice. Then, the slice reallocation process works into two attempts: 1) the first one tries using unallocated resources available in the physical infrastructure (line 3 in Algorithm 2)). In this scenario, unallocated resources can be added to the slice facing an increasing demand (line 4 in Algorithm 2). 2) the second attempt tries to create free-up resources to be added to that slice. In this step, the infrastructure searches for idle resource underutilisation in the remainder of the slices instances (line 5 in Algorithm 2). If there are idle resources in other slices, the infrastructure can reallocate the resources among them. If one of the attempts presents a positive answer, the slice can access more bandwidth to serve that increasing traffic demand, *i.e.*, the service migration. Otherwise, the migration will occur using the currently available bandwidth. The allocation cost is calculated within the allocateBandwidthForSlice process based on slice and link variables. The overhead costs in the reconfigurations are based on the proposed model described in Equation 6.1 that is further discussed in Chapter 6. The operators from Equation 6.1 were omitted in Algorithm 1 to increase the clearness of the algorithm. In Algorithm 2, slice and link variables are objects with attributes that describe, among others, the operational costs to proceed with operations from Equation 6.1.

4.5 End-to-end Slicing and Vehicular Network Support

The MobFogSim extension developed in this work focuses on E2E network slicing, *i.e.*, network slices that can cover all the network domains, and vehicular network support. The module in the simulator that handles that functionality is *Slice*. In addition, pre-

viously existing classes, namely *FogDevice*, *MobileDevice* and *APDevice*, were modified too. The modified classes are highlighted in grey in Figure 4.4. In fact, MobFogSim now allows access points (represented by *APDevice*) to fragment their resources, making them distributable among network slices. Besides, the *MobileDevice* module uses the *groupId* attribute to identify the mobile device as part of a group of devices having similar requirements. Based on this identifier, that device can access a specific network slice. In its turn, a network slice can serve one or more groups of devices.

In more detail, network slice configuration works as follows. The *Slice* module defines the amount of resources to allocate to each network slice in proportional or absolute terms. It does so by taking into consideration: i) the set of fog/cloud nodes, together with their respective resource offerings (storage and processing), as specified in the *FogDevice* module; ii) the set of links and access points along with their bandwidth and latency offerings, as defined by the *APDevice* module; and iii) the demands (*e.g.*, latency, bandwidth, processing power) of mobile devices, represented by the *MobileDevice* class. The allocation of network resources (*i.e.*, bandwidth) to each network slice is done through traffic reservation. Each network slice reserves an amount of bandwidth from a set of links. This reservation is defined as the number of packets exchanged in that slice which have the right to travel through that link. In other words, a slice with 30% of the bandwidth of a given link is entitled to send up to 30% of the packets transmitted over that link. In this MobFogSim extension, the transport and access network makes that resource reservation, which supports an end-to-end slice. This approach guarantees a resource reservation for each slice but does not optimise the use of idle (*i.e.*, unutilised) resources between network slices. A solution, supported by the network slicing concept and developed simulator, is dynamic resource reallocation, which was discussed in Section 4.4.

Finally, aiming to support device-to-device connectivity, in this new release of MobFogSim, the mobile devices (implemented by the *MobileDevice* class) also incorporate similar wireless connectivity features as presented in the *APDevice* module. The simulator reuses that implementation to establish direct communication between nearby devices. This one-hop communication avoids the dependency on the fixed infrastructure (*e.g.*, an access point/roadside unit) to connect two nearby devices.

4.6 Scalability

As introduced in Section 1, the 5G/fog computing-based networks can be high-density networks. The density of those networks depends on the place and time and can sometimes reach hundreds or even thousands of devices in a given area. One of the contributions of this work is the improvement of MobFogSim's scalability. The final aim is to let the tool simulate more realistic and denser networks than before while taking the same amount of processing time and using the same amount of resources.

Once MobFogSim extends CloudSim and iFogSim, on the one hand, extending those simulators allowed the building of one tool on solid foundations, exploiting their well-validated implementations and results. On the other hand, however, this choice caused MobFogSim to inherit the scalability issues of those simulators. Improving the simulator

scalability fixes a relevant issue inherited from MobFogSim predecessors while keeping all their advantages.

MobFogSim, as well as the two simulators that it extends, is an event-driven simulator. These simulators build an event queue (named future queue) that stores all the events planned to be executed in the simulation. To improve the simulator's scalability, the proposed performance improvements focus on optimising the mechanism for managing the simulator's event queue. Every event has an action in the simulation, *e.g.*, user movement, service creation, service migration, slice allocation, and packet sending, among others. In the initialisation, periodic events, such as resource usage consults, are scheduled/inserted in the event queue before the simulation starts. Those events are always scheduled eventually, for example, every 2 seconds. The remaining events, which represent the vast majority, are scheduled dynamically as the simulation proceeds. In MobFogSim, events are objects of the *SimEvent* type. The *SimEntity* class is responsible for creating, scheduling, and processing the *SimEvents*, leveraging the contribution of the event queue.

In the simulator, the event queue is a priority queue implemented as a data structure of the Java *TreeSet* type. That structure enables the simulator to store the events in a sorted set and to make useful operations, *e.g.*, search, insert, and delete, in a competitive time complexity $O(log(n))$, where n is the number of elements in the queue. However, the overall performance of the simulator depends on the time complexity of its operations and the scale of its scenario. Simulations in MobFogSim start paying in terms of performance when the number of elements in the network increases, as well as the map size and the simulation time. In those situations, in fact, the size of the event queue grows, which decreases the performance of each queue operation. In special, the operations over the events queue are highly frequent as they are the core of event-driven simulators.

The approach proposed in this work to improve the simulator's performance is based on reducing the number of items in the event queue. In special, the simulator was modified to insert periodic events in the event queue only when necessary. Figure 4.6 illustrates the previous approach used to manage the event queue. As shown, periodic events (blue) are scheduled at the initialisation phase, which happens before the simulation starts, *i.e.*, the simulation clock starts counting. As the simulation starts, the events scheduled to happen at the time defined by the simulation clock are extracted from the queue and then processed. As an output of processing the extracted events, new events (non-periodic, illustrated as the grey ones in Figure 4.6) may be created and added to the event queue.

This work introduces some modifications to the event queue of MobFogSim to minimise the number of items throughout the simulation. Figure 4.7 illustrates the new management of events in the simulator queue. In this work, the periodic events are no longer scheduled all at once in the initialisation phase of the simulation. Instead, those events are dynamically added throughout the simulation. Specifically, these periodic events are stored in an additional structure to wait until the right time to be added to the event queue.

To reduce the number of items in the queue, this updated version of MobFogSim implements one similar concept of one sliding window protocol, as illustrated in Figure 4.7. Instead of scheduling the periodic events throughout the simulation time, these events are added only inside the sliding window. The simulation clock and the last non-periodic event

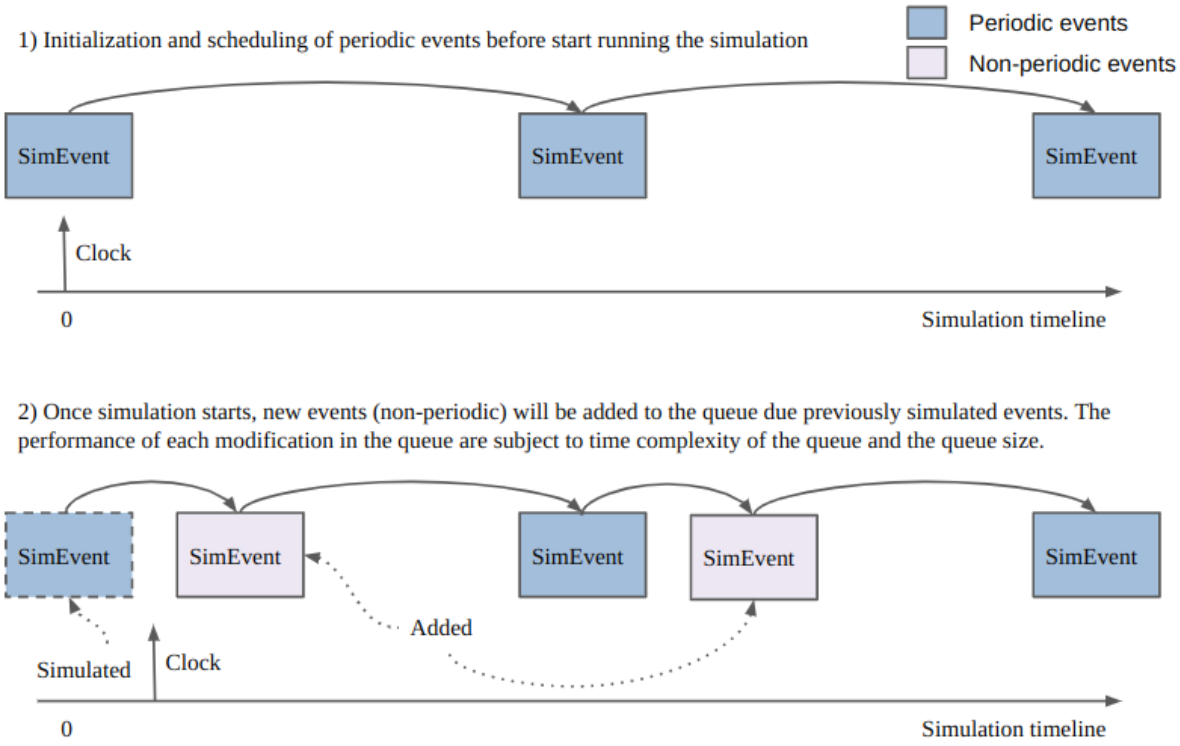


Figure 4.6: Example of inserting new events in the simulator's events queue based on the queue management of the previous version of MobFogSim. In that version, all periodic events are added to the event queue all at once in the initialisation.

bound that window. This mechanism does not insert periodic events in the queue after the last non-periodic one. In that way, periodic events that do not impact any event in the queue are not added at that moment.

However, aiming to keep the consistency between the two versions of simulations and the reliability of the results of the second one, the set of events and its order must be identical for the same simulation settings in the two versions of the simulator. In the previous version of the simulator, if different events were scheduled for the same simulation time, and one of them was a periodic event, that event was scheduled to be the first among them. The new version of the simulator maintains that same approach.

To execute that plan, once one non-periodic event is planned to be added to the event queue in a simulation time later than the last periodic event, as illustrated in Figure 4.7-3, all periodic events which are not in the queue and are planned to be executed at the time or earlier of that non-periodic event will be added to the queue. In special, these periodic events will be added to the event queue *before* adding the new non-periodic event. Adding the periodic events before adding the non-periodic one will ensure that the periodic one will be first placed in the queue in case two of them are scheduled at the same simulation time.

In general, the proposed modifications of that procedure ensure fewer events in the event queue while respecting the original order of the queue in the previous version of MobFogSim. The performance evaluation of MobFogSim in terms of scalability improvements is presented in Section 4.7 which also includes a comparison between results from a testbed and the proposed simulator.

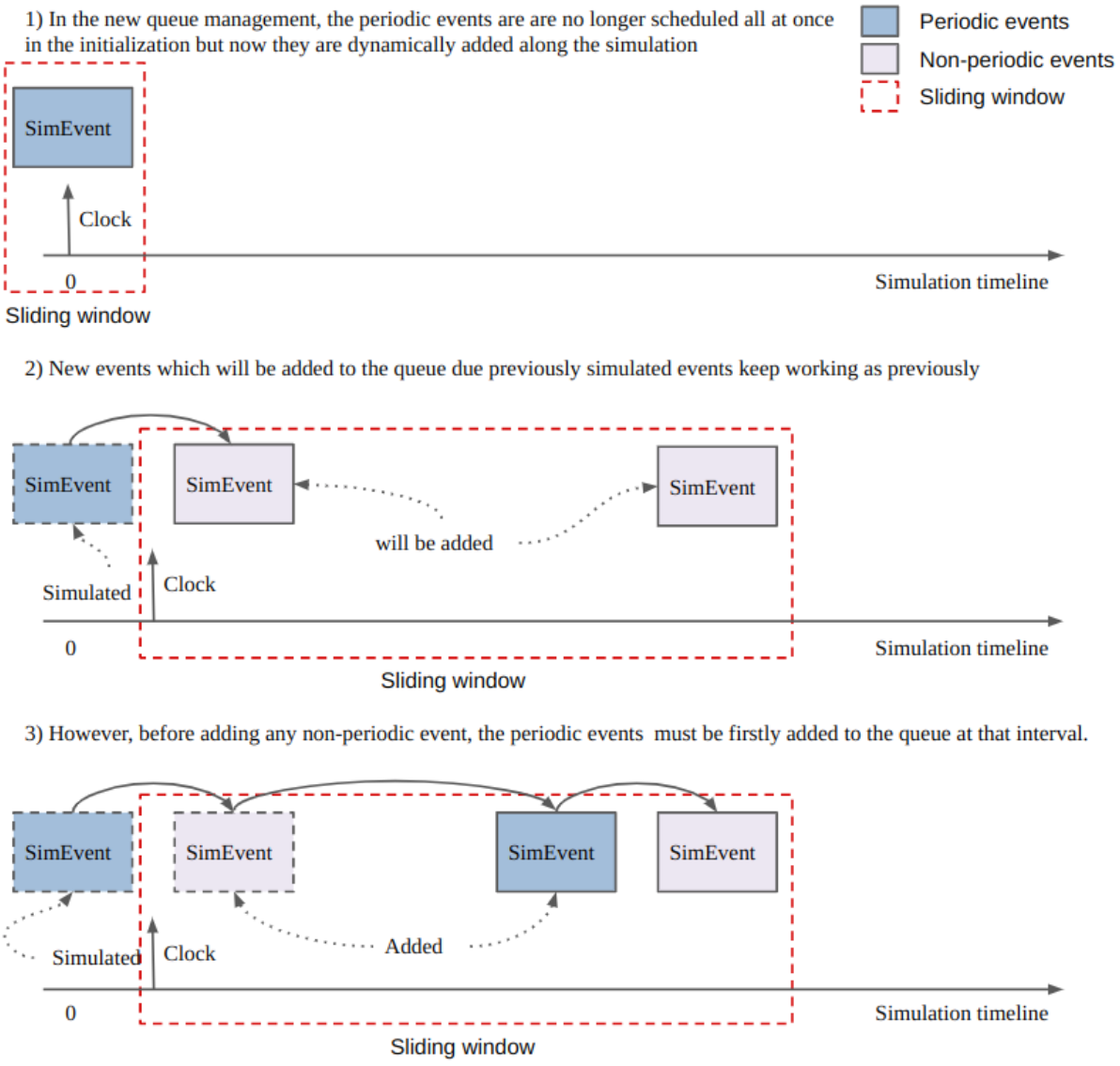


Figure 4.7: Example of inserting new events in the simulator's events queue based on the proposed queue management for MobFogSim. In this version, there are no periodic events later than the last previously non-periodic event scheduled in the queue.

4.7 Validation

Despite simulations can be a time and cost-effective way to evaluate resource management solutions in fog computing environments with mobile users, some aspects need to be carried out to improve its truthiness. In this section, it is presented how MobFogSim is validated. The simulation of service migration on edge computing architectures is the focus of the evaluation in this section. The experiments conducted in this work evaluated the virtual machine migration performance in terms of the required time to perform one migration, service downtime during the migration process, and the amount of data transferred during that process.

In this work, one real testbed was used to obtain seed values for supporting simulations in MobFogSim. Section 4.7.1 describes that real testbed scenario which is used to calibrate MobFogSim, *i.e.*, to obtain realistic input values for the next simulations. The main service migration results over the testbed are described in that section. Based on the testbed scenario, Section 4.7.2 presents the equivalent simulation setup in MobFogSim. Then, the results from the testbed are compared with those from MobFogSim to validate it. The validation of the remainder of MobFogSim features described in this chapter is further discussed in other chapters of this thesis, *e.g.*, network slicing in Chapter 5. Scalability aspects of MobFogSim are evaluated in Section 4.7.3.

4.7.1 Results Over the Testbed

This section describes the testbed infrastructure that is used as input for the MobFogSim validation. Also, this section reports the main container migration results obtained from the experiments over the testbed.

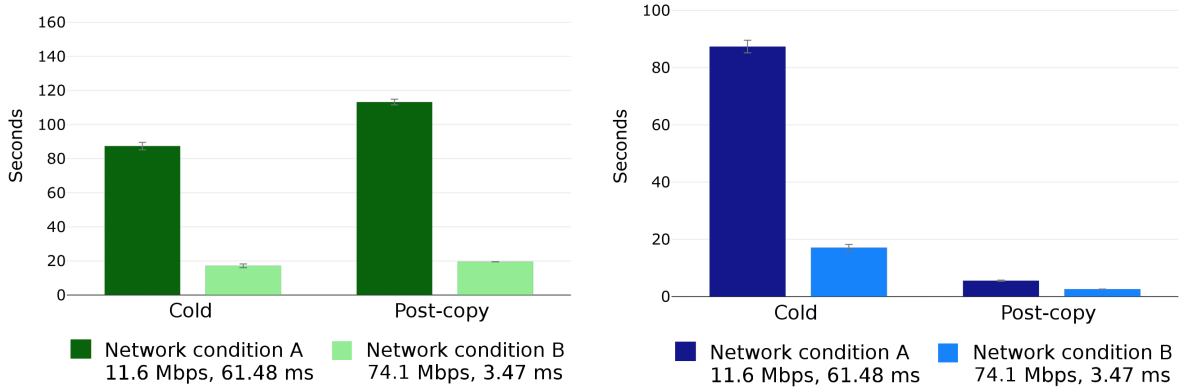
MobFogSim is validated by comparing simulation results with those obtained from a real testbed presented in [121] and [123], where fog services are implemented as containers. The testbed settings were used as input parameters for MobFogSim simulations.

In the testbed configuration, the user's device was an ASUS Zenbook UX331UN notebook, and the fog nodes were represented by two Raspberry Pi 3 Model B devices. Two network conditions were emulated in the testbed: (i) a *slower network* (Condition A) presenting, on average, a latency of 122.95ms and a throughput of 11.34Mbps; (ii) a *faster network* (Condition B), presenting a latency of 6.94ms and a throughput of 72.41Mbps. Aiming to obtain values for the slower network, a fixed device connected through Ethernet and a smartphone connected to the Internet through 4G/LTE were used. For the faster network, instead, it was employed two fixed computers belonging to a bridged LAN and located 1km far apart. All testbed settings are presented in Table 4.1. Further details about the testbed can be found in [121] and [123].

Despite the existence of many VM migration techniques, *e.g.*, cold, pre-copy, post-copy, and hybrid, MobFogSim currently simulates migrations only according to the cold and post-copy techniques. As a result, these are the only two techniques considered in this work. The experiment performed five runs for each combination of migration technique (*i.e.*, cold, post-copy) with network conditions (*i.e.*, A, B). In what follows, the results were analysed in terms of (i) total migration time, *i.e.*, the overall time required

to complete container migration; (ii) downtime, *i.e.*, the time interval during which the container is stopped for migration; and the service is therefore not available to the client; and (iii) total volume of data transmitted during migration. All the following results are presented with a 95% confidence level.

Figure 4.8(a) depicts the total migration times. These times for post-copy migration are always longer than those for cold migration, even though these two techniques transmit similar amounts of data overall. This result can be explained as follows. In cold migration, memory pages are transmitted from the source to the destination node without the need for any request from the latter. Instead, post-copy migration was implemented in [121] and [123] according to the "lazy migration" variant, which means that the infrastructure transfers memory pages on the source node only upon request of the lazy pages daemon running at the destination. Thus, the time to perform such requests, which are not present in cold migration, increases the overall total migration time of the post-copy technique. This difference between cold and post-copy migrations is particularly noticeable under network Condition A, where RTT between the fog nodes is considerably higher than that of Condition B.



(a) Total Migration Time in the testbed.

(b) Downtime in testbed.

Figure 4.8: Migration time and downtime under conditions A and B in the testbed.

As shown in Figure 4.8(b), downtimes are the main difference between the two migration techniques. In cold migration, the container is frozen/stopped throughout the whole migration procedure. As a result, downtime for cold migration is the highest among the migration techniques and even coincides with the total migration time. Cold migration downtimes may be unacceptable for most services, especially under low-throughput conditions (*e.g.*, Condition A). Post-copy migration, instead, is one of the "live" migration techniques. It means that the container is running while most of its state is being migrated to the destination node. In particular, post-copy migration first stops the container and transfers at the destination only the execution state (*i.e.*, the CPU state, the content of registers), whose size is negligible with respect to memory pages. It resumes the container at the destination and transmits all the memory pages while the container is up and running. Therefore, downtimes for post-copy migration are significantly lower than those for cold migration, as depicted in Figure 4.8(b). Note that the size of the execution state for the container is 1.2MB, which is significantly smaller than the overall volume of data

transferred to the destination fog node.

Figure 4.9 illustrates the volumes of data transferred during migration. Cold and post-copy techniques transfer similar amounts of data, namely about 115MB. It is because both these techniques transfer each memory page only once, unlike pre-copy and hybrid migrations, which indeed transfer higher volumes of data [123]. Moreover, the amount of data transferred by cold and post-copy techniques is irrespective of the network conditions between fog nodes.

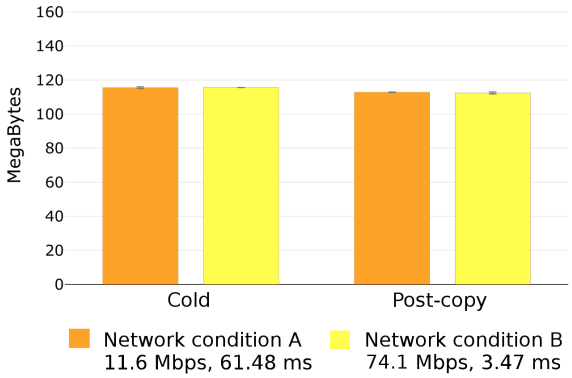


Figure 4.9: Amount of data transferred.

4.7.2 Results Over MobFogSim

In this section, the results of container migration on the testbed scenario presented in the previous section are compared with those simulated on MobFogSim. The preliminary experiments provided realistic values for the input parameters in MobFogSim. Aiming to simulate an environment as similar as possible to the testbed one, the simulation setup assumes fog nodes with homogeneous computing and network resources, as presented in the testbed. Although this experiment did not perform simulations in heterogeneous scenarios in terms of fog nodes' resources, it is supported by the simulator. For convenience, Table 4.1 reports these parameters in alphabetical order along with their values. Names in brackets are the actual variable names used in MobFogSim to indicate those parameters. The results are made over 30 simulations for each combination of migration technique (*i.e.*, cold, post-copy) with network condition (*i.e.*, A, B). Simulation results are analysed and compared with those from the real testbed in terms of (i) total migration time, (ii) downtime, and (iii) network usage in terms of (a) total volume of data transmitted during migration and (b) link usage. The results are presented with a 95% confidence interval.

In addition to the input parameters provided by testbed experiments, some complementary input values were assumed as part of the simulated environment. These simulation settings are described as follows. The simulated scenario assumes a square 10 x 10 km map with 144 uniformly distributed fog nodes. Each fog node is connected to one access point, which reaches up to 500 m of signal coverage. Each simulation assumes a uniformly distributed random direction for the user's mobility. The user is supposed to cross the map at a constant speed (20 kmph) until she/he reaches the opposite map edge. However, the simulations end once the user finishes her/his first container migration process. The

Table 4.1: List of input parameters and their values based on the testbed experiments for simulation in MobFogSim.

Parameter	**Value**
Client execution speed (*mips*)	2901 MIPS
CoAP request size (*tupleNwLength*)	87 B
CoAP response size (*tupleNwLength*)	54 B
Disk usage of client (*size*)	4 MB
Disk usage of server (*size*)	412 MB
Maximum speed of client (*mips*)	46534 MIPS
Maximum speed of server (*mips*)	3234 MIPS
Number of instructions executed by client (*tupleCpuLength*)	966 million
Number of instructions executed by server (*tupleCpuLength*)	2439 million
One-way latency between client and server (*UplinkLatency*)	4.78 ms
One-way latency under condition A between servers (*lat*)	61.48 ms
One-way latency under condition B between Servers (*lat*)	3.47 ms
RAM requirement of client (*ram*)	49 MB
RAM requirement of server (*ram*)	128 MB
Server execution speed (*mips*)	281 MIPS
Throughput from client to server (*upBw*)	13640 kbps
Throughput from server to client (*downBw*)	13363 kbps
Throughput under condition A between servers (*bw*)	11612 kbps
Throughput under condition B between servers (*bw*)	74148 kbps

migration point policy was defined as a fixed point. The migration process starts once the user reaches the migration point, which is defined at 40 meters from the access point coverage boundary. The container destination in the migration process is chosen based on a greedy approach. The migration strategy assumed in this evaluation selects the fog node with the Lowest Latency among a set of 10 candidate fog nodes that are present in the user's path. Aiming to evaluate a more flexible environment, it is evaluated three different execution state sizes transmitted during migration: 1.2 (which is the actual size from the testbed results), 6.0, and 12.8 MB. Table 4.2 summarises the above settings of the simulated environment.

Table 4.2: List of input parameters and their values assumed for the settings of the validation scenario in MobFogSim.

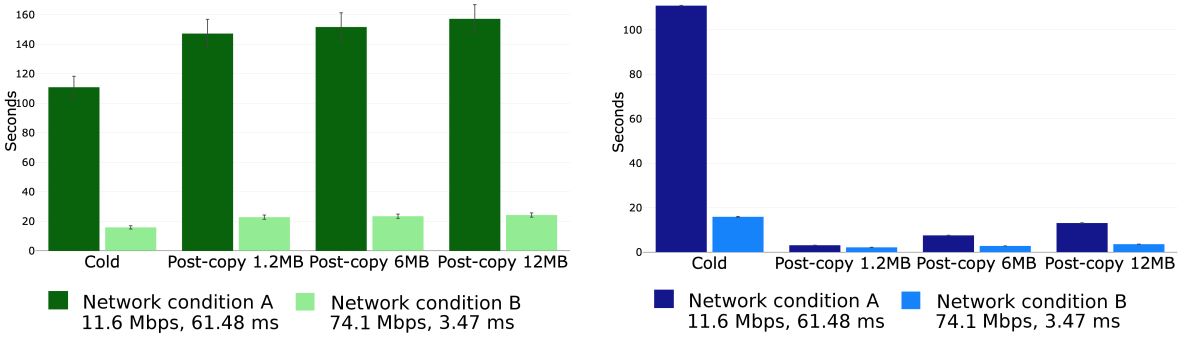
Parameter	**Value**
Container's execution state size in live migration process	1.2 MB, 6.0 MB, and 12.8 MB
Access point coverage (radius)	500 m
Number of fog nodes	144
Density of fog nodes per access points	1:1
Migration strategy	Lowest latency
Migration point policy	Static (40 m)
User's speed	Constant (20 kmph)

Figure 4.10(a) presents the total migration times in the simulated scenarios. Similarly to the testbed results presented in Figure 4.8(a), post-copy migration, in all its variants, presents higher values than cold migration under the correspondent network conditions. As explained in Section 4.7.1, this result is because post-copy migration transfers memory pages only upon request, and the time for such requests increases the total migration time. Going into detail, under condition A, both the simulated and the testbed post-copy scenarios present a migration time about 30% longer than that of cold migration. However, the simulated environment presents average values that are 25% higher than those in the correspondent testbed scenario. Under condition B, both the simulated cold and post-copy migrations present similar results to the testbed.

In the particular case of post-copy migration, the size of the execution state transmitted in that process does not significantly impact the migration time. Increasing the execution state size from 1.2 to 12.8MB resulted in an increment of about 6% in the migration time under both network conditions, A and B.

Based on these simulation scenarios used as a starting point for comparisons, in general, the total migration time presented by MobFogSim tends to be consistent with the testbed results for both cold and post-copy migrations.

Another relevant metric for evaluating the migration techniques is the downtime, which is presented in Figure 4.10(b). Similar to the testbed results presented in Figure 4.8(b), the downtime of cold migration is higher than that of post-copy. It even coincides with the migration time presented in Figure 4.10(a) under both conditions A and B. Even though the execution state size does not significantly impact the migration time, this parameter strongly influences post-copy migration in terms of downtime. Assuming an execution state size of 12.8MB, the simulations suggest a downtime about 420% higher under condition A and 66% higher under condition B if compared to migrations with an execution state size of 1.2MB.



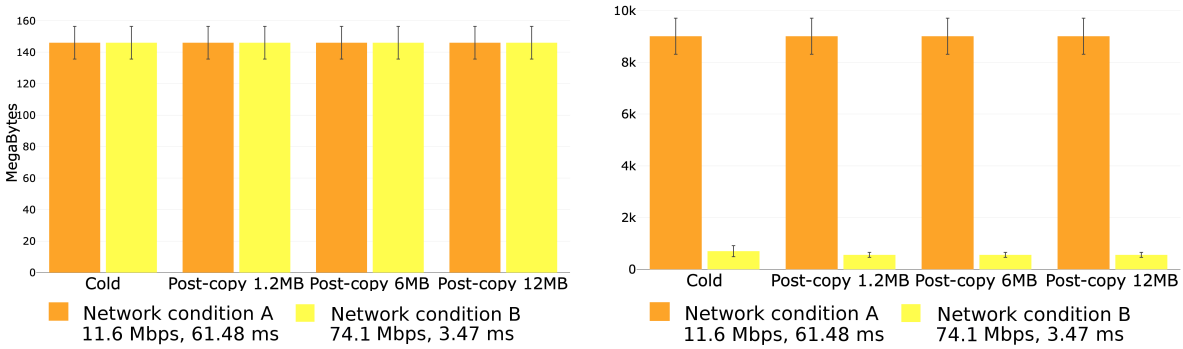
(a) Total migration time in MobFogSim.

(b) Downtime in MobFogSim.

Figure 4.10: Migration time and downtime under conditions A and B in MobFogSim.

Aiming to complement the evaluation of the migration techniques used in the scope of this work, it also presents a performance evaluation in terms of (a) the total volume of data sent between two fog nodes over the migration process and (b) link usage based on the amount of data transferred through the network and the latency between the fog nodes. Figure 4.11(a) presents the amount of transferred data between the fog nodes.

The simulation scenario presented a volume of transferred data close to 150MB, which is about 30MB higher than the testbed environment. As presented in the testbed results (see Figure 4.9), the volume of data sent is stable under both network conditions. Besides, as in the testbed, both migration techniques transfer a similar volume of data (nonetheless, the total migration time for post-copy is higher due to the time necessary to request memory pages). As a complementary metric, Figure 4.11(b) presents the link usage, which is defined in this work as the amount of data sent multiplied by the link latency between the source and destination fog nodes. This metric clarifies the difference between the two network conditions, A and B, in terms of efficiency. On average, condition A presents an efficiency of around 1200% higher than condition B.



(a) Amount of data transferred.

(b) Link usage (data transfer x link latency).

Figure 4.11: Network usage under conditions A and B in MobFogSim.

4.7.3 MobFogSim Scalability

Previously in this chapter, Section 4.6 presented a discussion regarding the scalability improvements developed in the MobFogSim life cycle. In addition, this section presents the results of the proposed modifications. In this section, it is presented a performance evaluation between MobFogSim 1.0 [121] in perspective of the second release of the simulator, MobFogSim 2.0 [70].

For what concerns the assessment of the improved simulator scalability, the experiments consider two metrics: i) the total execution time, measured in minutes; ii) the maximum RAM consumption, measured in GB, on the computer running the simulations.

The experiments perform tests using different loads on the simulator. Namely, 20, 50 and 80 vehicles per simulation were used to evaluate the enhanced version of the simulator (MobFogSim 2.0). On the other hand, it did not consider the 80-vehicles case to test the previous version of MobFogSim (version 1.0) because of the time consumption and computational resource limitations. The machine used to run the tests contains a 14-core Intel(R) Xeon(R) CPU E5-2660 v4, with an operating frequency between 2.0 and 3.2GHz and 226GB of RAM. Results are the average among 30 simulation runs, considering a confidence interval of 95%.

Figure 4.12 presents the results obtained after testing MobFogSim scalability. The

first metric evaluated (see Figure 4.12 (a)) concerns the time required for the complete execution of the simulations, from the allocation of resources to the presentation of results. As depicted, the performance of version 2.0 of the simulator is considerably better than version 1.0. For a scenario with 20 vehicles, version 1.0 presents a total execution time of 121 minutes on average, whereas, in version 2.0, this time is reduced to 48 minutes, representing a decrease of 60%. When considering 50 vehicles (*i.e.*, 2.5 times more vehicles than the previous case), version 1.0 performed about 2.8 times worse, going to 342 minutes of execution on average. On the other hand, version 2.0 of the simulator increased execution time by 2.25 times, going to 108 minutes. For the 80-vehicles scenario, version 2.0 had an average runtime of around 180 minutes.

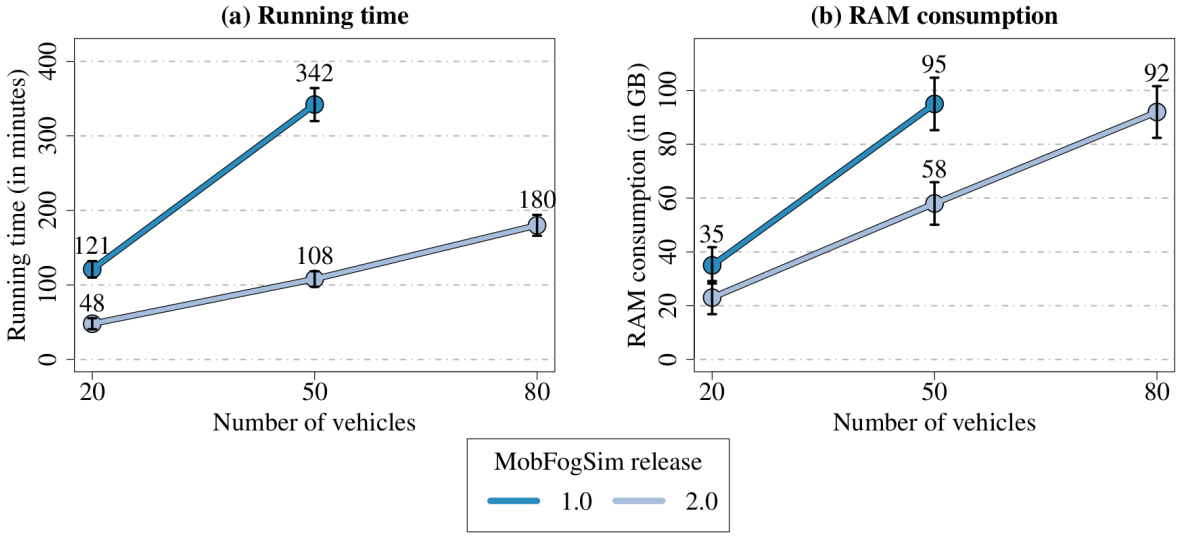


Figure 4.12: Scalability of MobFogSim in terms of runtime and RAM consumption.

Another metric considered in this evaluation concerns the maximum consumption of RAM. Long times to execute a simulation are not necessarily a limiting factor for the feasibility of the experiments. On the other hand, high consumption of RAM can limit the number of machines capable of running simulations. Figure 4.12 (b) presents the results in terms of RAM consumption. The superiority of version 2.0 of the simulator is evident. The maximum RAM consumption in simulations with 20 vehicles is, on average, 35GB for version 1.0, while version 2.0 consumed 65% of this value, namely 23 GB. For scenarios with 50 vehicles, version 1.0 has an increase of 2.7 times, going up to 95GB, whereas version 2.0 has a 2.5x increase, reaching 58 GB. Version 2.0 still consumes 92GB of RAM for the scenario with 80 vehicles. Note that, for similar consumption levels, close to 92 GB, there was a 60% increase in the number of vehicles between the two versions, namely from 50 vehicles in version 1.0 to 80 in version 2.0.

The presented results show significant improvements to the scalability of the simulator if compared to its previous version. MobFogSim can now run simulations with the same number of vehicles as in the previous version, using fewer RAM resources and taking less time to execute. Conversely, by fixing the amount of consumed resources and/or the target execution time, it is now possible to simulate more realistic scenarios involving a higher number of vehicles. Considering that VANETs are often highly dense networks,

this is a fundamental result for all those who want to test their VANET-related solutions in MobFogSim.

4.8 Chapter Conclusions

MobFogSim simulator has been developed to fill one gap in the literature. Testbed infrastructures may not be the best option to validate solutions for user mobility support in network slicing scenarios because they present scalability, flexibility, and monetary limitations. The proposed simulator is unique in the state-of-art. This work presented several contributions to extending CloudSim and iFogSim, two well-known and well-validated simulators compared to its previous versionswere. Specifically, MobFogSim implements end-to-end network slicing in static and dynamic ways, which includes the pool of storage and processing resources from all the layers in the network infrastructure. Also, one realistic user mobility support, which enables the simulator to access real data from traffic datasets, was developed. Moreover, this work added the modelling of vehicular scenarios to the simulator, including the possibility for vehicles to behave as fog nodes that share their resources with nearby users. Furthermore, this chapter presented the significant improvement made in the scalability of MobFogSim. That improvement allows the simulator to run more complex scenarios involving a higher number of network elements while consuming fewer resources.

All these enhancements make MobFogSim a complete tool for evaluating solutions involving, among others, fog offloading, fog service migration, device mobility, and network slicing. Furthermore, the simulator was validated by comparing its results with testbed data. The experiments suggest that simulated and real results are equivalent. MobFogSim is available as Free Software.

Once the validation environment is developed and validated, further studies of network slicing performance can be carried out on MobFogSim. Some of those studies, which include the development and evaluation of resource policies for dynamic network slicing, are presented in the following chapters of this thesis.

Chapter 5

Dynamic Network Slicing for Mobile Users

Modern computer networks like 5G, 6G and beyond must simultaneously serve users with diverse and sometimes conflicting requirements. In special, mobile users boost the challenges by increasing traffic variations and requiring continuous connectivity. In such a context, network slicing arises as a supporting technology for resource management in such networks. In a nutshell, network slicing is a way to create several virtual networks (slices) on top of a physical infrastructure. Each slice will be designed to serve the specific requirements of its users, such as low latency and high bandwidth or ensure a minimum threshold of reliability.

In this scenario of focus on user mobility support, the proper resource allocation at the beginning of the slice life cycle is not enough to provide the necessary connection between users and their services. In that case, the infrastructure should be able to quickly adjust the slice configurations according to demand variations caused by users mobility. The state-of-art works do not present further evaluations regarding the performance of dynamic slicing. This chapter aims to fulfil that gap.

The previous chapter of this thesis described the development of the MobFogSim simulator, a simulation environment to support the evaluation of solutions for resource management in fog computing environments, including service migration, network slicing, and support for users' mobility. This chapter investigates some performance evaluation tests regarding dynamic network slicing. In special, the contribution of this chapter explores the evaluation of 1) the impact of dynamic network slicing over static slicing allocation, 2) the role of dynamic slicing in serving mobile users' requirements, 3) how slices with different resource requirements impact the resource management and, 4) how vehicular networks can explore the benefits of dynamic slicing.

The contribution of this chapter is published along four peer reviewed articles [66], [67], [68], and [70]. For the sake of clarity and convenience, this chapter might introduces a different version of the content present in those articles.

The remaining of this section is organised as follows:

- Section 5.1 presents one evaluation of network slicing as supporting technology to improve the service migration process. The scope of that evaluation includes static

and dynamic slicing performance;

- Section 5.2 expands the previous evaluation by including mobile user support as a requirement for previous slicing scenarios;
- Section 5.3 introduces the performance of dynamic network slicing as a supporting technology for vehicular networks;
- Section 5.4 presents this chapter's conclusions and final remarks.

5.1 Network Slicing for Service Migration

This section evaluates network slicing as a supporting technology for improving the service migration process. Experiments were built to evaluate different slice resource allocation approaches in that context. Section 5.1.1 describes the performance of static slicing allocation, and Section 5.1.2 presents the results for dynamic slicing allocation.

The implementation of fog computing presents several challenges. Device mobility is one of them. When a device/user moves through the road while connecting to different APs, the communication distance to the fog service may increase. In such a scenario, the benefits of using fog nodes may no longer be available, *e.g.*, lower latency. A possible approach to solve this issue is to migrate the user's service across the fog infrastructure to keep it always close enough to the mobile user[122]. That process is known as the follow-me cloud.

These service migration and resource management approaches face a complex scenario due to the significant diversity and heterogeneity of users and service requirements in fog environments. Mobile users may move at different speeds (*e.g.*, on foot or by car) and according to different mobility patterns. In addition, some services may present strict requirements in terms of latency; others may prioritise throughput, while others may have specific hardware needs. In such a scenario, some users may require a higher priority for managing their service migration, managing their migration speed, or requiring specific destination node(s). Traditional network infrastructures cannot guarantee all of these, sometimes concomitant levels of security, performance, reliability, privacy, and/or other particular requirements for each user group.

Based on the implementation of network slicing, as present in Section 2.2, each slice can be tailored to fulfil the requirements of a particular group of users and/or services. Furthermore, network slices can be dynamically reconfigured and reallocated to address the changing needs of users and avoid underutilisation of network resources (*i.e.*, via dynamic network slicing). In this section, network slicing is presented as a supporting technology for service migration in fog. In this context, network slicing is used to introduce priority for some groups of users in the migration process. Furthermore, dynamic network slicing is used to improve resource management and avoid idle resources. Two allocation approaches are considered, static and dynamic, and different scenarios were evaluated in terms of available and demanded resources. The content of this section is based on the work present in [66].

5.1.1 Static Network Slicing

This section evaluates the impact of static network slicing on service migration to support mobile users within a fog computing environment. In static resource allocation, scale-up or scale-down capabilities are not allowed. MobFogSim models network, storage, and computing resources. However, this section focuses on bandwidth slicing. The analysis of the other aspects, such as the impact of storage and computing resources on the slicing process, is present in Section 5.3. As a result, all mobile users received the same amount of computing and storage resources in the form of containers. However, the experiments considered different distributions of users and network resources (*i.e.*, bandwidth) in the simulations. It is worth mentioning that while this section focuses on static network slicing, Section 5.1.2 instead considers the case of dynamic network slicing.

As presented in Section 2.2, the resources assigned for one slice are granted until the end of that slice's life cycle in the static resource allocation approach. However, resource reallocations are not available. In the face of the migration process designed in MobFogSim, the request for more bandwidth resources (lines 6 and 7 in Algorithm 1) is not available for that static allocation approach.

Simulation setup

Figure 5.1 presents the physical and logical architectures used in the experiments. The physical architecture of the fog computing environment is composed of two fog nodes connected by routers in the access and transport networks. Users are connected to the fog computing infrastructure through wireless access. Two network slices, namely Slice 1 and Slice 2, are created on top of the shared physical infrastructure.

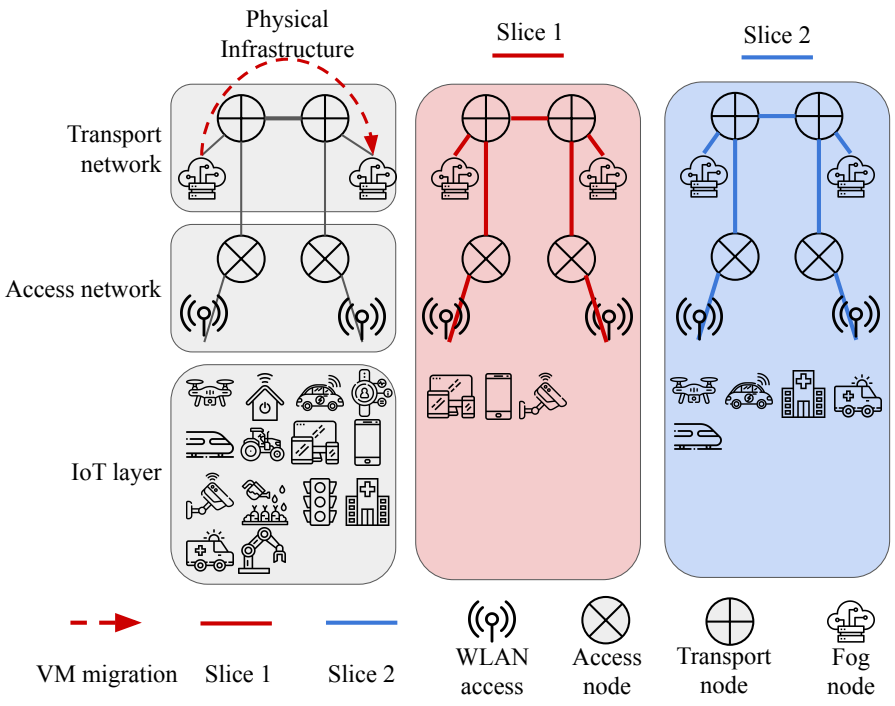


Figure 5.1: Physical infrastructure and network slices used in the evaluated scenarios.

All the scenarios considered consist of 20 users whose mobility caused 20 container migration from one fog node to another. All 20 users started from the same point on the map and moved in the same direction and speed. As a result, all migrations started simultaneously. The simulation finishes when the container migrations are finished for all the users. The input values used in the simulations were the same considered in the preliminary experiments that are summarised in Table 4.1 and Table 4.2 in Section 4.7.2. Despite these points, the simulation scenario is only evaluated on the faster network, the number of fog nodes is restricted to 2, the number of users is increased to 20, and all users present the same constant direction and speed mobility pattern.

In the experiments, 20 users were distributed in three different ways between the two network slices. Specifically, it is assumed the following three distributions of users between Slice 1 and Slice 2, respectively: (i) 2 against 18; (ii) 4 against 16; and (iii) 8 against 12. Furthermore, the experiments considered three different distributions of network resources between the slices. The following three distributions of network resources between Slice 1 and Slice 2 were built, respectively: (i) 10% against 90%; (ii) 30% against 70%; and (iii) 50% each. Combining the distributions of users and resources was considered in nine different scenarios in the simulations. It is worth mentioning that the container's data transmission was performed using packets up to 1MB.

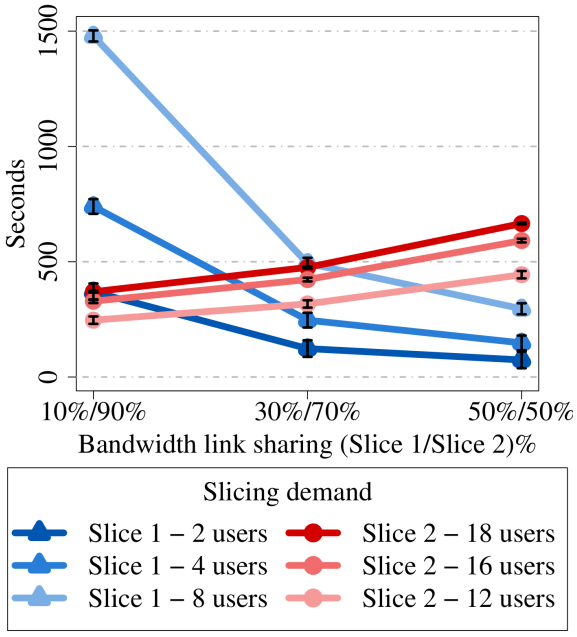
For each scenario, the following three metrics were considered: (1) total migration time, (2) percentage of link utilisation throughout the simulation, and (3) the amount of data that could be transmitted while the link is idle. Link utilisation (2) is calculated based on the average amount of data sent through that link and is expressed in terms of the percentage of the link capacity. The link used in this measure connects the two fog nodes and is used to transmit the containers' data during migration. The bandwidth available in that link is 74Mbps based on the testbed setup defined in Section 4.7. Finally, the potential amount of data not transmitted (3) is calculated based on the time that the bandwidth resources are allocated but not used. The container migrations were performed according to the post-copy technique. Twenty-five simulations were performed for each scenario. Results are shown with a confidence interval of 95%.

Results

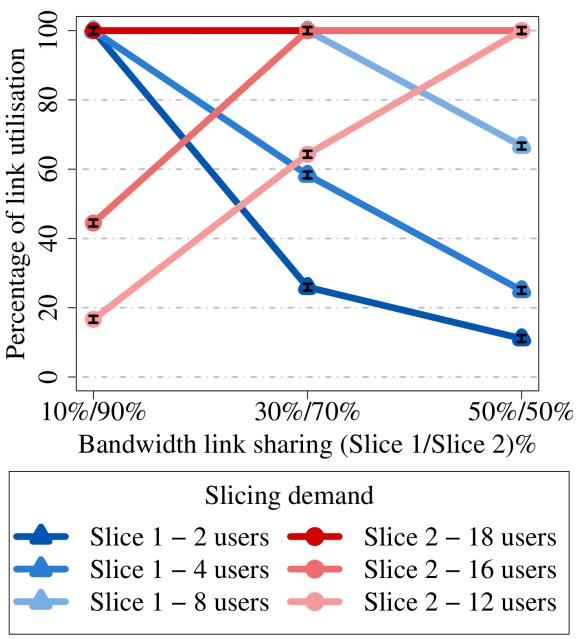
Considering that each container in the simulations has a data size of 128MB, 20 users simultaneously request migrations, which leads to 2.56GB of data being transmitted in parallel. Using post-copy migration in a traditional network, container migration would take 369s on average. In this context, it is possible to discriminate the traffic for different users through network slicing and introduce some priority for them. Based on the simulation scenarios, the impact of the balance between users' requirements and resource allocation on the migration process was analysed.

Figure 5.2(a) presents the average total migration time. The blue lines represent the performance of Slice 1, while the red lines are related to Slice 2. It is possible to notice how crucial the portion of bandwidth dedicated to each slice is. In the first scenario, where only 10% of the bandwidth was allocated to Slice 1, the average migration time for four users in Slice 1 (739s) is almost twice that of 16 users in Slice 2 (328s). The performance

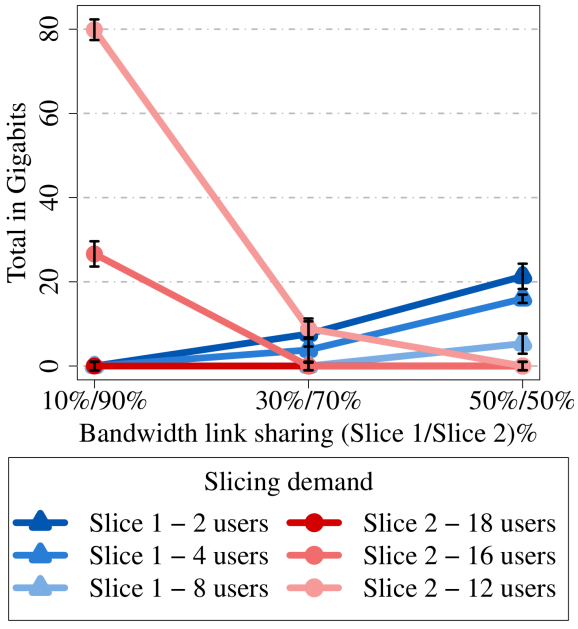
of Slice 1 with eight users is even worse: the average migration time in that slice (1479s) is almost six times that of the remaining 12 users in Slice 2 (246s). The 10%/90% scenario presents an unfair distribution of network resources. However, the average migration time tends to be more propitious to Slice 1 as the bandwidth is more equally shared. In the 30%/70% scenario, the migration times for Slice 1 and Slice 2 are more similar. In the last scenario (50%/50%), migration times for Slice 2 are between 130% and 900% longer than those for Slice 1 because fewer users belong to the last slice.



(a) Average migration time.



(b) Percentage of link usage.



(c) Amount of data not transmitted while the link was idle.

Figure 5.2: Performance of static allocation of network slices.

Figure 5.2(b) presents the results of link utilisation. Firstly, it is possible to note how links that show worse migration times in each scenario (see Figure 5.2(a)) use their entire assigned bandwidth. Those are indeed the links where the last container terminates its migration, which triggers the end of the simulation. Allocation of network resources is efficient when there is a high link utilisation (*i.e.*, no wastage of resources) and migration times meet users' requirements. However, the allocation may be inefficient in two other cases. In the first case, links are at their utilisation limit, but migration times are far above the required ones. An example of this case is that of 8 users assigned to Slice 1 with 10% of bandwidth allocated. The second inefficient case is that one in which network resources are idle (*i.e.*, not utilised after all migrations for that slice terminated) for a long period. Those resources, which could be leveraged to boost migrations for other slices, are indeed wasted. For instance, when 12 users are assigned to Slice 2 with 90% of network bandwidth, only 20% of those resources are actually used. Figure 5.2(c) can help to better understand this concept. It presents the amount of data that could be potentially transmitted while network links are idle. As already discussed, the scenarios where resources are not fairly balanced present a higher index of wasted bandwidth. This unfair balance is justified by the objective of decreasing the migration times for high-priority users. However, static slicing can not reallocate idle resources to other slices that need them. For instance, the scenario in which 12 users are assigned to Slice 2 with 90% of bandwidth achieves average migration times of 246s. However, once migrations for Slice 2 are finished, its network resources are not reassigned to Slice 1. It leads the average migration times for Slice 1 to be 1479s, while about 80GB of data could have been potentially transmitted for that slice.

The results in this section provide two insights. Firstly, network slicing can prioritise users and reduce the time it takes to migrate their containers, allowing them to perform more migrations and keep their services closer to them. Secondly, static network slicing causes an inefficient allocation of resources, which cannot be reassigned to other slices to boost their migrations. The following section presents an evaluation of how dynamic network slicing solves this problem.

5.1.2 Dynamic Network Slicing

This section analyses the impact of dynamic network slicing on service migration for mobile users in the fog. In dynamic network slicing, resources can be reassigned at runtime to other slices that request them. Combining service migration and dynamic slicing, the infrastructure can reallocate more bandwidth to slices which presents a higher demand. In that scenario, a request for resource redistribution among the slices can be made when service migrations are in progress, as presented in lines 6 and 7 in Algorithm 1.

As discussed in the previous experiment, static resource allocations are subject to do not match the levels of the allocated resources to their demand. As presented in Figure 5.2(c), one slice can receive more resources than needed, resulting in underutilisation. A worse case can also occur if the resource assigned to a slice is insufficient for its demand. In that scenario, the service migration can be harmed. In this section, dynamic resource allocation is analysed in the context of service migration. In this experiment, slice's idle

resources can be assigned to other slices which presents a higher demand.

Simulation Setup

The simulation setup defined for this set of experiments is the same as that reported in Section 4.7. The only difference is related to how idle resources are dealt with. In the previous section, the portion of bandwidth that is allocated to a slice is guaranteed to that slice for the entire simulation. In this section, instead, it is possible to reallocate resources on-demand, following the concept of dynamic network slicing. In this work, only idle resources are reallocated. Therefore, once all the migrations for a slice finish and the link becomes idle, that part of the bandwidth is completely reallocated to the other slice.

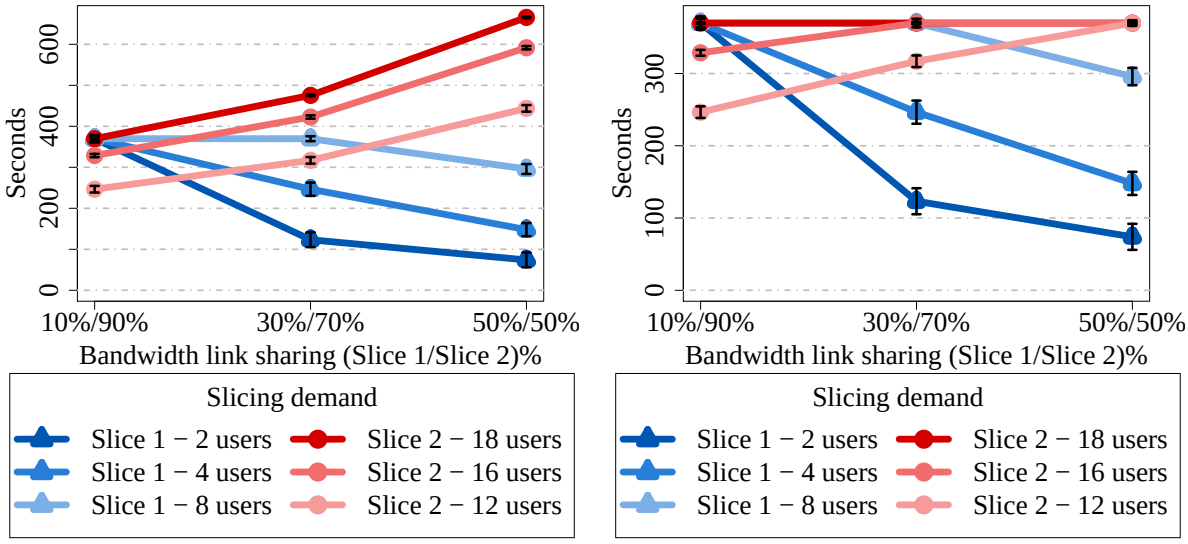
As described in Section 2.2.2, the process of reallocating resources among slices has computational costs that result in time overhead. That cost, which is set as a parameter in MobFogSim, depends on the characteristics of the physical and logical networks. For these experiments, that cost was defined as 1s. However, different values may impact overall performance.

The experiments in this section are organised into two parts. In the first one, users in Slice 1 are defined as higher-priority users. Based on that, once bandwidth becomes available in Slice 2, it is reallocated to Slice 1. The opposite does not occur. In the second part of the experiments, Slice 1 has no higher priority. Then, both slices can benefit from a reallocation of resources once these become idle. Results are relative to 25 runs per scenario and shown with a confidence interval of 95%. The post-copy technique was used to migrate containers.

Results

Figure 5.3(a) presents average migration times in slicing scenarios with unidirectional reallocations. In these experiments, Slice 1 receives the available bandwidth from Slice 2 once this resource becomes idle. This procedure helps Slice 1 in those scenarios in which, with static slicing, it presented a worse performance than Slice 2 (see Figure 5.2(a)). Scenarios where Slice 1 receives 10% of the resources and serves four and eight users, and the scenario in which it receives 30% of the bandwidth and serves eight users present average migration times around 50%, 25%, and 75% of those obtained with static allocation, respectively. Using dynamic slicing, the worst result for Slice 1 is limited to 369s compared to 1479s with static slicing. It is withdrawing idle resources from Slice 2 results in neither an improvement nor a worsening for that slice. Considering all scenarios, dynamic slicing leads to an average migration time of 406s, which is almost 15% less than the 469s of static slicing.

There are also some scenarios in which Slice 1 presents idle resources. Those resources could be used to improve the network performance for users in Slice 2. In this second group of experiments, Slice 1 also shares its idle resources with Slice 2. Figure 5.3(b) shows average migration times in slicing scenarios with bidirectional reallocations. Under these conditions, Slice 1 presents the same performance as that in Figure 5.3(a). However, also Slice 2 now receives a boost in its worst cases. Both slices are now limited to 370s in their worst cases. The average migration time for Slice 2 presents an improvement, being almost



(a) Average migration time using dynamic slicing to prioritise one slice.

(b) Average migration time using dynamic slicing to reallocate resources for both slices.

Figure 5.3: Performance of dynamic network slicing

15% less than that with the static allocation of resources. With bidirectional reallocation, link utilisation reaches nearly 100% for both slices. Therefore, dynamic network slicing achieves an efficient resource allocation, as it avoids under-utilisation of network links while reducing migration times for mobile users.

5.2 Network Slicing for Mobile Users

This section evaluates the network slicing allocation in a scenario with realistic user mobility patterns. Despite the performance of the resource allocation approach presented in the previous section, demand variations induced by user mobility may result in a different environment than the one previously studied. In such a complex scenario, the performance of dynamic network slicing needs to be further evaluated.

Based on the simulation setup presented in the previous section, additional scenarios were built to evaluate the performance of static and dynamic network slicing allocation using a realistic user mobility pattern in a larger-scale infrastructure [67].

As presented in the experiments in Section 5.1, the evaluated scenario consists of only two APs and two fog nodes. In addition, the mobility of the users was restricted to a constant direction and speed. In this section, a more extensive infrastructure is considered, and the user mobility is based on a realistic database of vehicles from Luxembourg (illustrated in Figure 5.4). Further details about the simulation setup are presented in the following sections.

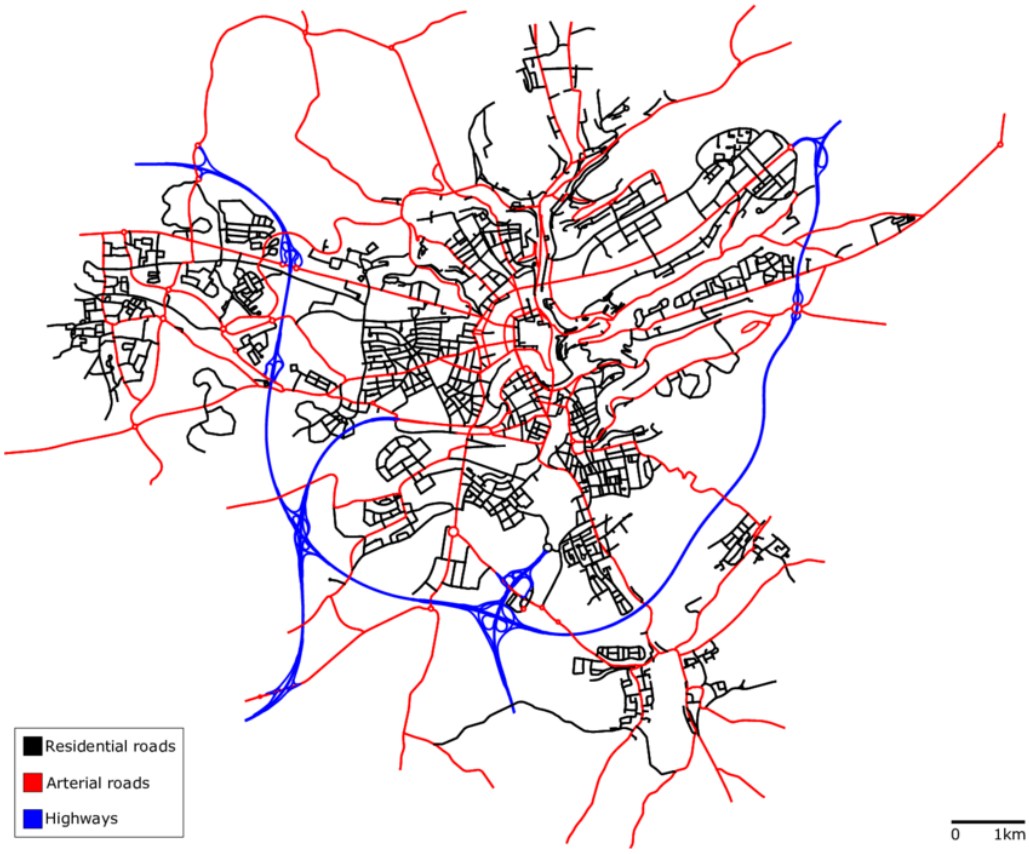


Figure 5.4: Luxembourg SUMO Traffic database topology. Source: [31]

5.2.1 Static Network Slicing

The experiments present in this section follow the same structure as those presented in Section 5.1.1. In the static slicing allocation, each slice receives a fixed portion of bandwidth during the whole execution of the simulation. Unlike the experiments presented in the previous section, the objective of this new experiment is to evaluate static slicing over demand variations. In this experiment, realistic user mobility is considered in a larger-scale infrastructure. Like the previous experiment, the static resource allocation was evaluated in different resource demand and availability scenarios. Details about the simulation setup are presented below.

Simulation Setup

The value of the input parameters considered in this experiment is the same as presented in Table 4.1. Additional values are presented in Table 5.1. The main contrast of the experiments in this section is the use of realistic user mobility patterns and considering a larger infrastructure.

In this experiment, the fog computing infrastructure has 16 fog nodes uniformly placed on a 5kmx5km map. 64 access points provide the connection between the users and the infrastructure. Based on the testbed setup defined in Section 4.7 a 74 Mbps bandwidth and 3.47 ms latency link connect the fog nodes. 2400 users are considered in this work.

Table 5.1: List of input parameters and their values assumed for the settings of the Slicing allocation experiments.

Parameter	Value
Container's execution state size in live migration process	1.2MB
Access point coverage (radius)	500m
Number of fog nodes	16
Density of fog nodes per access points	1:4
Migration strategy	Lowest latency
Migration point policy	Static (40m)
User's speed	Variable - real data

The user mobility was based on data of vehicles from Luxembourg SUMO Traffic (LuST) database [31]. Based on these 2400 users, 30 groups with 80 users were created. Each simulation used one of these groups. The simulation time is limited to one hour.

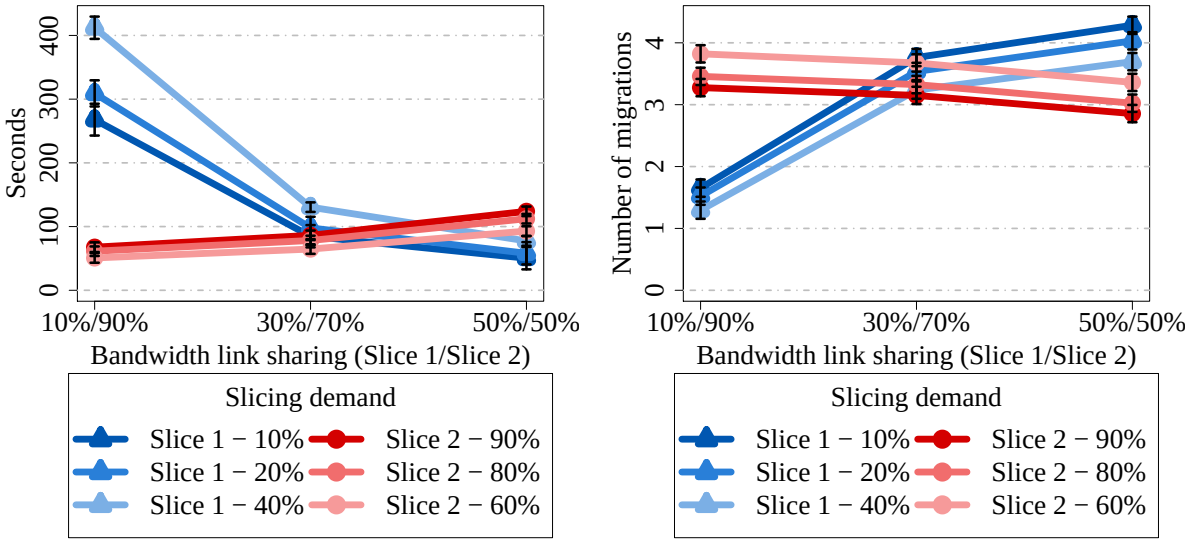
Similar to the experiments presented in Section 5.1, in this experiment, two slices, named Slice 1 and Slice 2, were created based on a static allocation approach. Each slice receives a fixed portion of the bandwidth available in the physical links. However, different values for resource allocation were evaluated. Like the previous experiment, three different levels of resources were assigned to each slice. Slice 1 received 10%, 30%, and 50% of the available bandwidth in the link, while Slice 2 received 90%, 70%, and 50%. Moreover, three different levels of user demand were evaluated. Slice 1 was evaluated serving 10%, 20%, and 40% of the 80 users considered in each simulation. Slice 2 served 90%, 80%, and 60% of the 80 users. Each level of resource availability was evaluated with each level of user demand in a total of 9 different scenarios for each slice.

The performance of static slice allocation is evaluated in terms of the following metrics: the number of migrations performed along with the user travel, the time required to commit each migration, and the latency perceived by the users. The results presented in the following section are calculated based on an average of 30 simulations. The considered confidence interval is 95%.

Results

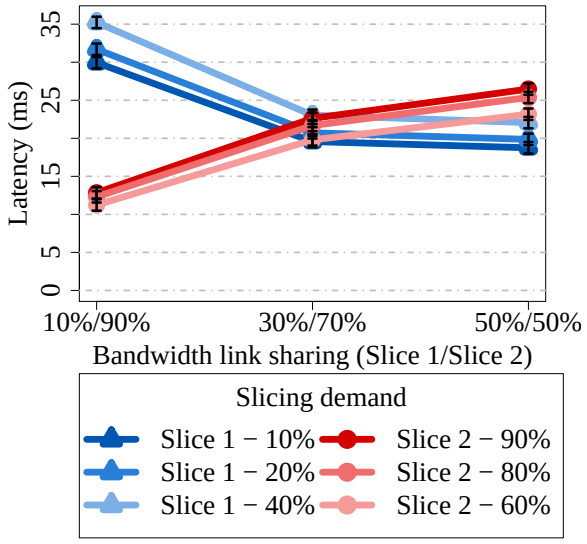
One of the main metrics used to describe the service migration performance is the time required to carry out one migration. Service migration time is highly affected by the amount of bandwidth available to proceed with that migration. Figure 5.5(a) shows the average duration of this migration process. In this scenario, as expected, both factors, resource availability and user demand, have impacted the migration process. However, in the experiments, resource availability presented a higher impact.

In a scenario with an equal distribution of bandwidth between network slices (50% for each), the slices with more users (higher demand) presented a worse performance. One slice serving 90% of users performs migrations about 240% slower than the slice which serves 10% of the users. In scenarios with an unequal resource distribution, slices with 10% of the total available resources can perform their migrations up to 400% slower than the slices that have 90% of the physical link bandwidth.



(a) Average duration of service migrations

(b) Average number of migrations



(c) Average latency

Figure 5.5: Simulation results of static network slicing

It is worth mentioning the presence of a non-linear performance in resource availability and user demand. It can be noted if the performance of a network slice with 90% of resources serving 90% of users and a network slice with 10% of resources serving 10% of users are compared. In this scenario, although the total demand is equivalent to the amount of available resources, each area of the map may receive different demands. In this context, users placed in less crowded areas have more available resources *potentially*. However, a user assigned to a slice with 10% of the resources, even in a case where he/she does not find concurrency in his/her region, he/she will be limited to access only 10% of the network capacity. Another user in the same conditions but being able to access 90% of the network resources will obtain a better performance.

The time required to perform a migration directly impacts other metrics in the net-

work. Figure 5.5(b) presents the average number of migrations performed by each vehicle during the simulations. Assuming that a finite path restricts the users' mobility, there is a relation between the two metrics: the time required to perform a migration and the number of migrations performed. As shown in Figure 5.5(a), the network slices with access to higher bandwidth resources presented more migrations. Slices with fewer users requiring resources also presented better results.

The migration time associated with the number of migrations that the infrastructure can provide to the user, along with its path, results in how close one service can be placed to its mobile users. In that context, a higher number of migrations means that service placement can be adjusted as the user moves around the map. As shown in Figure 5.5(c), the network slices that performed the higher number of migrations reduced the latency for their users. It is noted because the migration avoids the latency increases due to the users moving away from its service. As seen in the previous figures, the migration capacity is strongly related to Slicing accessing more bandwidth for that process. In the figure, the network slices with the fewest resources had an average latency up to 350% higher than the resource-rich slices. In a scenario with equal distribution of resources (50% for each network slice), the variation in user demand resulted in a performance up to 40% higher.

The network slicing allows the distribution of the physical resources among different network slices, each one being able to meet different requirements. As presented in the experiments, both resource availability and user demand impact the service migration performance. In that context, an unbalanced resource distribution may be assumed as one of these two actions: the desired approach or a side effect. As presented in the introduction of this work, some services require specific resources. Network slices that receive more resources introduce privileges or priorities to their users compared to other network slices. That process boosts the performance of those higher-priority slices at the cost of harming the performance of the remaining ones. Dynamic factors, such as user mobility, may increase that unbalance. The static resource allocation policy, evaluated in this section, does not allow changing on these network configurations if needed. An alternative to that scenario is dynamic network slicing, in which the resources allocated to one slice can be reallocated over time to better serve users' demands.

5.2.2 Dynamic Network Slicing

Based on the experiments for evaluation of the static slicing allocation presented in the previous section, new experiments were carried out considering the capacity of the network to redistribute its resources over time. In this experiment, as the demand for each network slice changes over time, the NFV Management and Orchestration (MANO) system can change the shape of the slices to support that demand variation successfully. In this new scenario, only idle resources can be reallocated to the other slice if that slice requires it.

Simulation Setup

Unlike the previous experiment for static resource allocation, this experiment allows the slices to request idle resources from other slices to boost their service migration. In this context, the *requestSliceReshaping* function, presented in Algorithm 1, reallocates idle

bandwidth during the migration process. As discussed in Section 2.2.2, the process of resource reallocation has an overhead presented in terms of time and computing costs. In this context, as discussed in Section 4.4, that temporal cost is defined as an input parameter in MobFogSim and is set as 1s in this experiment. The remaining simulation setup considered for this experiment uses the same parameters presented in Section 5.2.1, in terms of resource availability, resource demand, network settings, and user mobility.

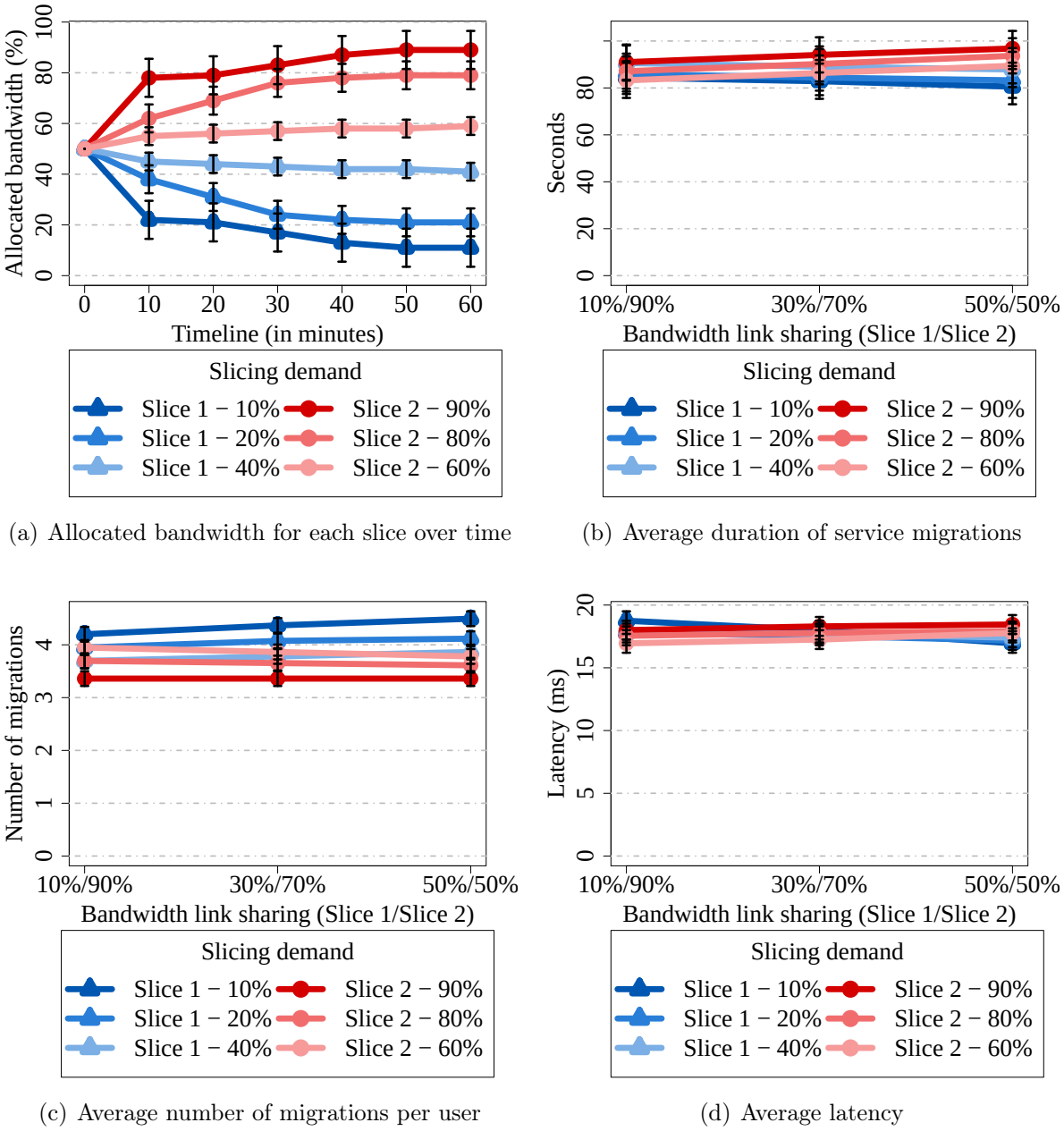
Results

Aiming to illustrate the network resource reallocations, Figure 5.6(a) shows the percentage of assigned bandwidth for each network slice over time. In the figure, each slice received half of the available bandwidth on the physical infrastructure. In this evaluated scenario, as the users connect to the infrastructure and, consequently, require resources to migrate their services between the nodes, the demand in the network slices changes according to the number of users served by that slice. Since each network slice started the experiment with the same amount of resources (50%), the slices that served more users consequently presented higher demand levels. Unlike the static scenario, these network slices can require idle resources from other slices. The figure presents, in all scenarios, that the reallocation policy resulted in a resource redistribution that tends to be equivalent to each network slice demand.

The resource allocation policy adopted in this work only reallocates idle resources. Slices with higher demand tend to request more resources than to release them. Despite presenting an equal resource distribution, it removes the priority introduced to some network slices. As discussed, that priority factor may be desired for particular slices in some scenarios.

The result of that resource redistribution presented by the allocation policy adopted in this work, presented in Figure 5.6(a), impacts other metrics used to evaluate the network performance. Assuming that each slice will receive resources equivalent to its demand, the duration of each migration tends to be similar. Figure 5.6(b) illustrates the performance of each network slice based on that metric. In this experiment, the performance between the network slices did not exceed the 30% among them. It is different from the performance observed in the previous experiment, which presented results up to 400% different. Furthermore, it is noted that there is a significant improvement in the performance of the network slices that previously received fewer resources if compared to the static allocation scenario. In the previous scenario, such slices were limited to the low percentage of resources allocated to them. In the dynamic scenario, resource-poor slices could access more resources, mainly in regions with less competition. In concurrent regions, the presence of idle resources is rarer.

As a consequence of the duration of the migrations, the number of migrations performed, shown in Figure 5.6(c), as well as the average latency noted for each user, shown in Figure 5.6(d), also presented similar performance.



(a) Allocated bandwidth for each slice over time

(b) Average duration of service migrations

(c) Average number of migrations per user

(d) Average latency

Figure 5.6: Simulation results of dynamic network slicing

5.3 Network Slicing in Vehicular Networks

This section evaluates the performance of dynamic end-to-end network slices in vehicular networks. This experiment evaluates the network resource management performing vehicular clouds' resources as part of the end-to-end slice.

Vehicles are gradually evolving from simple means of transportation to smart devices. The current trend is to equip vehicles with sensors, actuators, compute and memory resources, and communication capabilities. Smart vehicles can form complex networks, which are known as Vehicular Ad-Hoc Networks (VANETs) [94]. In VANETs, communications can be Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), or Vehicle-to-Everything (V2X). In the latter case, vehicles can exchange information with any other

device, such as pedestrian handhelds, drones, or smart bicycles, to name a few.

In this context, VANETs are challenging to manage due to the high mobility, high density, and high heterogeneity characteristics of its users. Integrating VANETs with other networking paradigms can effectively address some challenges. In this section, three network paradigms are assumed: vehicular networks, edge computing and network slicing. The experiments presented in this section evaluate how end-to-end network slicing can influence the performance of fog service migration in VANETs while considering several use cases that differ on where fog nodes are placed in the network.

In this section, three fog-assisted VANET scenarios are evaluated. These scenarios differ in the way fog nodes are deployed. The *fog_only* scenario assumes fog nodes to be deployed only at the edge of the fixed network infrastructure. The *vehicular_only* scenario considers fog nodes resources to be provided by nearby vehicles instead. Finally, the *hybrid* scenario combines the previous two. Figure 5.7 illustrates the consider scenarios.

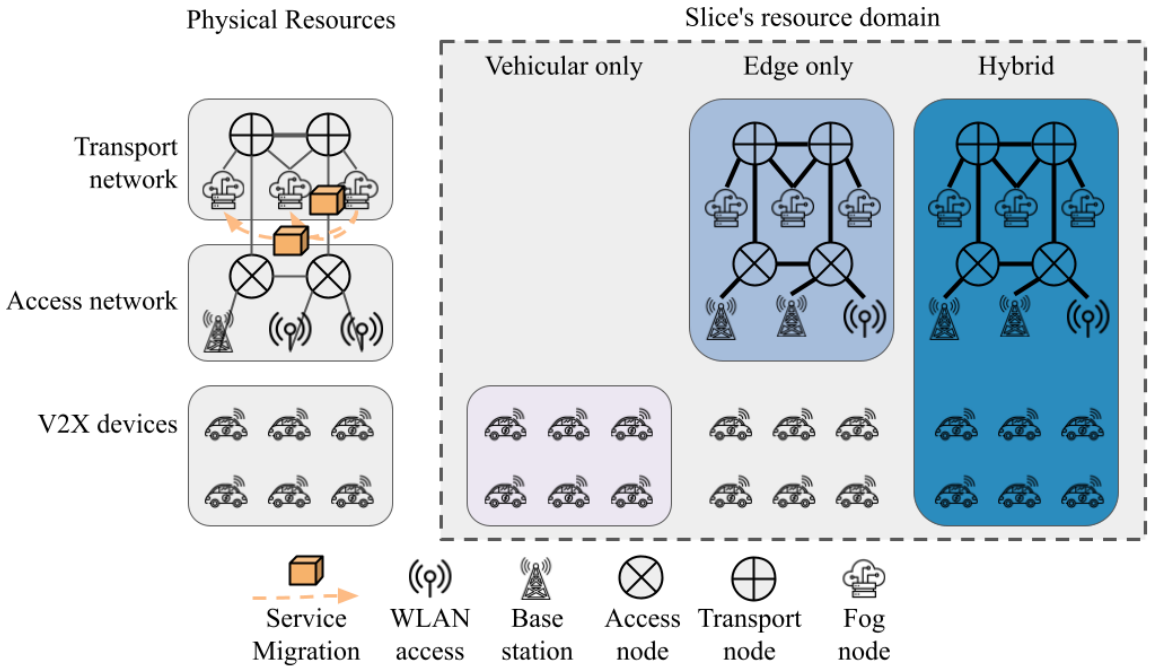


Figure 5.7: Resource domains considered in this experiment: vehicular only, edge only and hybrid.

The metrics considered in this first part of the evaluation are network latency, bandwidth, location of fog service placement (vehicle or edge infrastructure), number of fog service migrations, and number of slice reconfigurations. Section 5.3.1 presents the experiment setup, and Section 5.3.2 presents the results of this experiment.

5.3.1 Simulation Setup

Scenarios that are typically found in smart cities were used to validate the simulator. Each scenario simulates the behaviour of the network in 60 minutes and involves 80 vehicles. Vehicle mobility patterns were taken from LuST [31], which provides data from Luxembourg. From that city, a 5x5km map is considered. That map was split into nine

zones, as shown in Figure 5.8. The vehicles inside one zone compose a vehicular cloud and can share their available resources or request additional ones from nearby vehicles. On average, the vehicles stay 4:26 minutes into one zone and visit, on average, 4.31 zones along their trip. Overall, 2400 vehicles from the LuST database were selected, which allows running 30 instances of each scenario. The average speed of those vehicles is 22.5kmph in an average 18:23 minute trip.

For what concerns the network parameters, realistic values were chosen based on the testbed setup defined in Section 4.7 as input to the simulator. Table 5.2 summarises these values. Note that all the scenarios this work considers assume a client-server application model. Specifically, the client application always runs on vehicles, which are the client devices from Table 5.2. Instead, the server application is the fog service running on fog nodes. As explained before, fog nodes can be i) nodes located at the edge of the fixed infrastructure – which this work defines as *fixed fog nodes* from now onwards – (*i.e.*, *fog_only* scenario); ii) other vehicles in the vicinity of client devices (*i.e.*, *vehicular_only* scenario); or iii) a combination of the previous two approaches (*i.e.*, *hybrid* scenario). It is worth highlighting that, in the proposed simulations, only one fog node hosts a given fog service at a given time. In other words, those scenarios wherein a fog service consists of sub-components that can be deployed on different fog nodes were left for future work. The network architecture consists of 80 vehicles, 16 fixed fog nodes, and 64 access points positioned at the network edge. Half of the connections between fog nodes randomly present a bandwidth of 74Mbps, whereas the other half has a bandwidth of 148Mbps. These network links have an average one-way latency of 3.47ms. The V2I

Parameter	**Value**
Client application processing requirement	[2901, 5802] MIPS
Client device processing capacity	3234 MIPS
Server node processing capacity (Fog)	6468 MIPS
Server node processing capacity (Vehicle)	[3234, 6468] MIPS
Client application storage requirement	4 MB
Server application storage requirement	412 MB
Client application RAM requirement	49 MB
Server application RAM requirement	128 MB
Request Packet size	87 B
Response Packet size	54 B
One-way latency of access point	4.78 ms
One-way latency in links connecting fog nodes	3.47 ms
One-way latency between Vehicles	4.78 ms
Access point bandwidth	13.6Mbps
Access point coverage (radius)	500 m
Vehicle communication range (radius)	500 m
Fog node bandwidth	[74.1, 148.2] Mbps
Vehicle bandwidth	[74.1, 148.2] Mbps

Table 5.2: Parameters and their values used in the simulations. Some values are based on the infrastructure presented in [121].

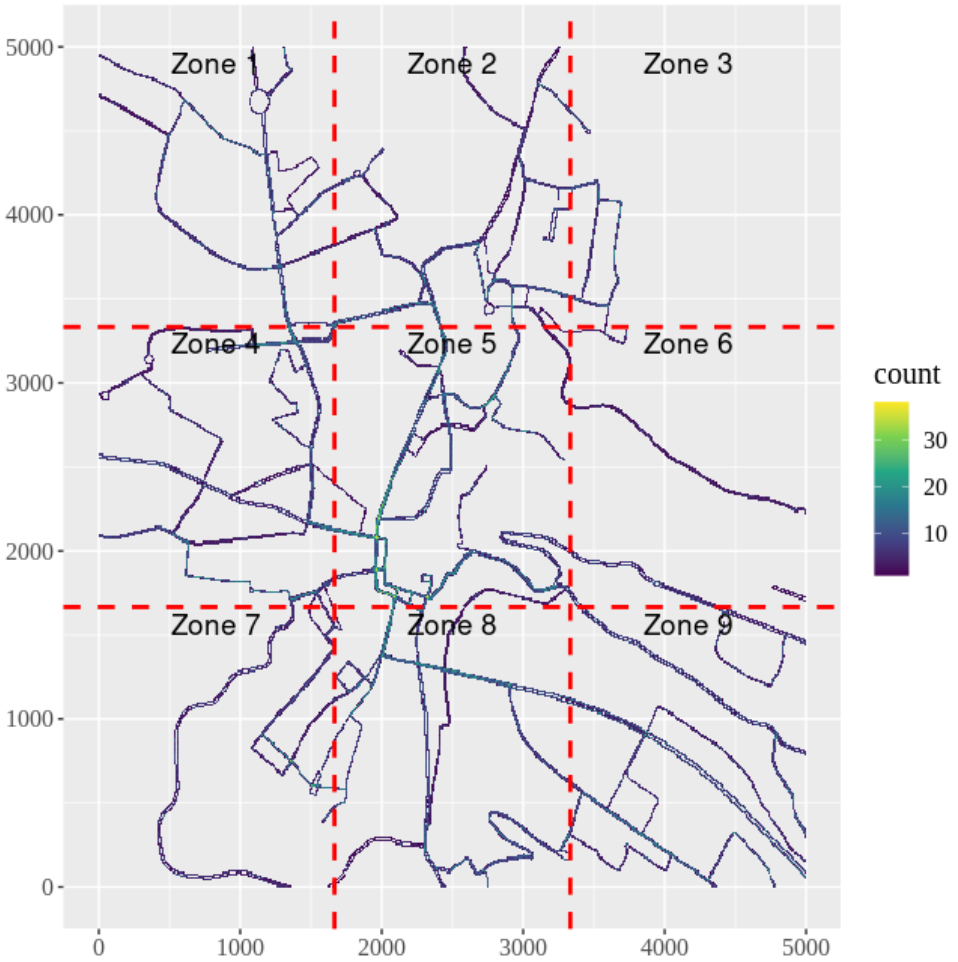


Figure 5.8: Map of Luxembourg and its zones created for this work.

and V2V communication range reaches up to 500m of signal coverage, respecting 802.11p theoretical and practical limits [12]. Fixed fog nodes have a processing capacity of 6468 million instructions per second (MIPS). Half of the vehicles potentially act as fog nodes and, therefore, have the same capacity as fixed fog nodes. Instead, the other half of the vehicles have a capacity of 3234 MIPS. The processing requirement of fog services changes between two values (2901 and 5802 MIPS) over time for a period of two minutes.

Regarding the slice settings, three slices were created over the simulated physical infrastructure. Each network slice focuses on a specific aspect of the network: i) *Slice 1* prioritises processing capacity and RAM; ii) *Slice 2* prioritises bandwidth, and iii) *Slice 3* prioritises latency. The slices were allocated in the above order, following a simple priority-based policy. Based on such a policy, slices that are placed in the first positions of the allocation queue have more chances to get the required amount of resources. When resources that best meet the requirements of a slice have been assigned to that slice, the infrastructure moves to the next slice in the queue. It is worth mentioning that the slice allocation order can have an impact on some results. This work focuses on priority-based ranking to schedule the slice allocation. The considered policy can indeed harm lower-priority slices, which risk starving and provide lower performance to their users. Different scheduling approaches, changing the allocation order or providing a global solution may

present a different performance than the one provided in this work. Nonetheless, an analysis of the impact of slice ordering deserves further investigation and is left for future work. Besides, MobFogSim can be further extended by researchers and leveraged to evaluate other slice allocation policies, such as coordinated or machine-learning-based allocations. The validation scenario results in an average among 30 simulation runs, considering a confidence interval of 95%. The results related to the use of network slicing in fog-assisted VANETs are discussed below, in Section 5.3.2.

5.3.2 E2E Network Slicing in Fog-assisted VANETs

Figures 5.9 and 5.10 present the fog resource distribution among the considered slices in the case of fixed-fog and vehicular-fog resources, respectively. As shown, Slice 1, which prioritises processing capacity, receives half of the available fog resources in both cases. Slice 2 and Slice 3, instead, receive 30% and 20% of the resources, respectively. Besides, it is possible to note how, in the *fog_ only* and *vehicular_ only* scenarios, the slices only access resources from their respective domain. On the contrary, the *hybrid* scenario can allocate resources from both domains.

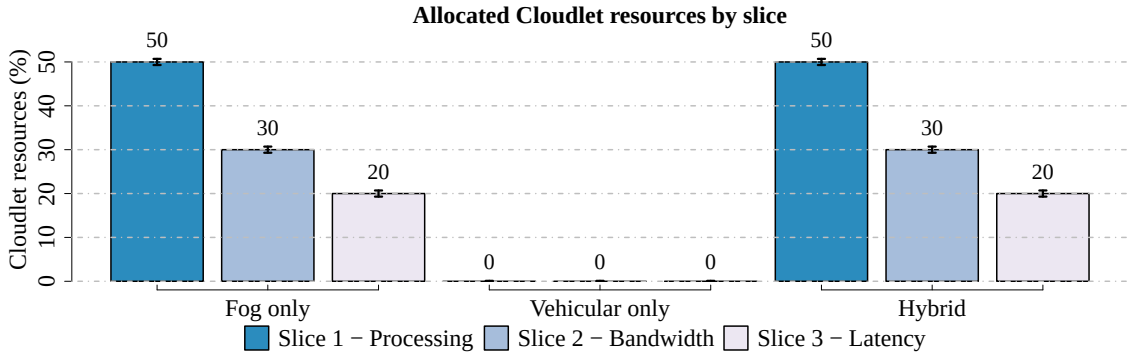


Figure 5.9: Fixed-fog resources allocated to slices.

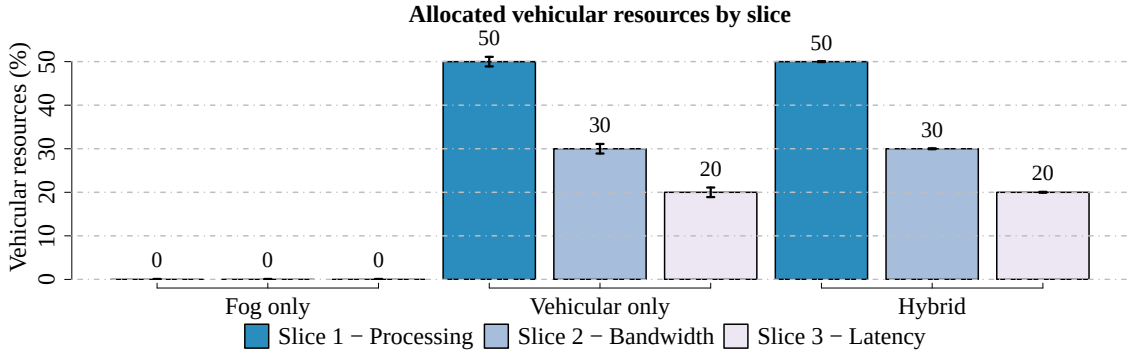


Figure 5.10: Vehicular-fog resources allocated to slices.

Figure 5.11 presents the bandwidth distribution among the slices. It is possible to note that there are no different levels of bandwidth availability among the different scenarios (*fog_ only*, *vehicular_ only*, and *hybrid*). Those values are in scale to the number of fog nodes allocated to the slices. For example, slices in the *hybrid* scenario have more fog

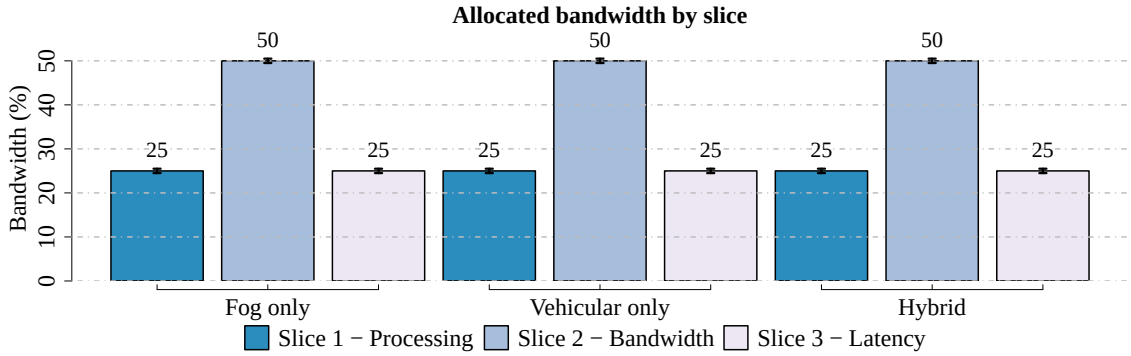


Figure 5.11: Bandwidth allocated to slices.

nodes (vehicles + fixed fog nodes) than in the other scenarios. However, its links present, on average, bandwidth equivalent to other scenarios. Despite that equivalence in terms of bandwidth among the scenarios, Slice 2 received twice more bandwidth than the remaining slices in all three scenarios. That result is expected considering Slice 2 prioritisation of bandwidth.

Based on the above resource distribution among the slices, MobFogSim can simulate how the network performs in terms of fog service placement and fog service migration to support device mobility. Figure 5.12 and Figure 5.13 present the relative number of fog services that are allocated on fixed fog nodes and vehicles, respectively. In the *fog_ only* scenario, as expected, no fog services are placed on vehicles. The opposite happens in *vehicular_ only* scenarios, where all services are placed on vehicles. The *hybrid* scenario takes advantage of choosing the best fog node to place the service among both vehicles and fixed fog servers. In the *hybrid* scenario, Slice 1 mainly selects fixed fog nodes to host its fog services (42.8% remaining in vehicles) due to the higher resource availability in that domain (some vehicles have lower processing capacity than the fixed fog nodes, as presented in Table 5.2). Due to the bandwidth being equivalent in both vehicles and fixed fog nodes, Slice 2 places its fog services in a balance between the two domains (52.3% in vehicles and 47.7% in fixed fog nodes). Finally, Slice 3, which prioritises low latency, has its fog services mostly on vehicles.

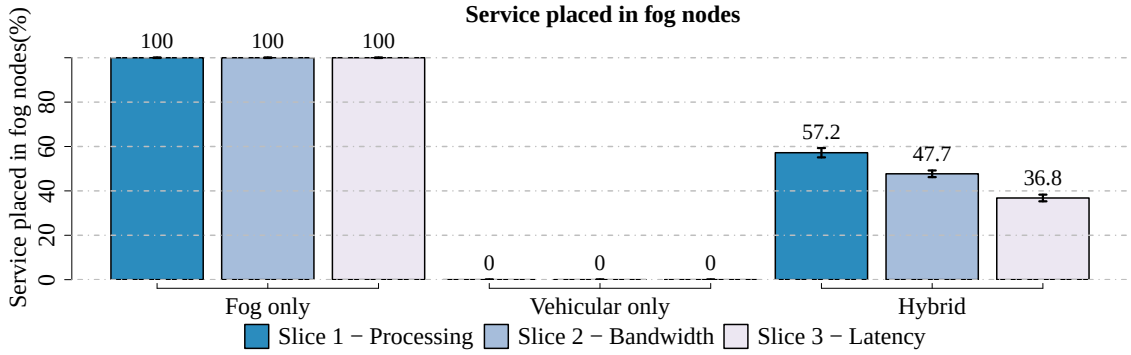


Figure 5.12: Percentage of fog services placed on fog nodes.

Once the resource availability of each slice and the placement of the services in the network were discussed above, the following discussion introduces how much resource

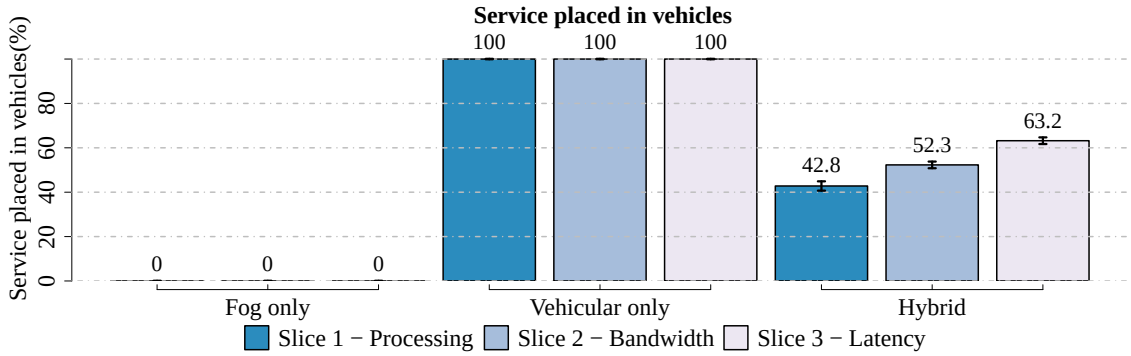


Figure 5.13: Percentage of fog services placed on vehicles.

each slice and each network layer, *i.e.*, vehicles or fog nodes, are using from its available resources. Figure 5.14 presents the percentage of the network resources that each slice used through the simulation in the evaluated scenarios, *i.e.*, *fog_ only*, *vehicular_ only*, and *hybrid*. In this context, resource usage is related to the processing required by the application in the face of the processing capacity of the host. Based on these data, Figure 5.14 indicates how much fog resources are used by each slice, considering its resource availability and demand. As expected, the hybrid scenario presents the lowest resource usage rates for its slices compared to the other approaches. It is justified once the hybrid approach has vehicles and fixed nodes at the network's edge available to its slices. On the other hand, the *fog_ only* and *vehicular_ only* approaches presented higher resource usage rates because they are limited to only one kind of resource source. Among the slices, the Network Slice 1, which received half of the available fog resources in the simulation, presented the lowest resource usage in the three evaluated approaches if compared to other slices, which may indicate that Slice 1 resources were sufficient to serve its users. On the other hand, the lowest usage rate in the hybrid approach may indicate that a significant part of the Slice 1 resources was underutilised.

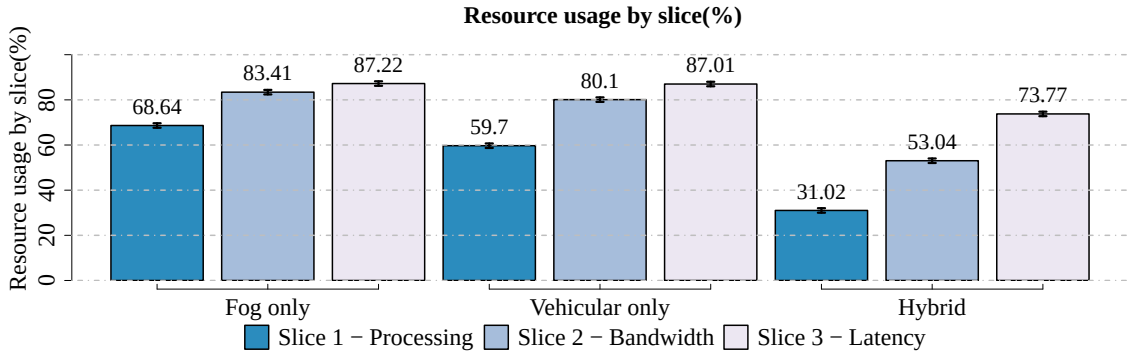


Figure 5.14: Percentage of resource usage in each slice.

Despite the static allocation approach adopted in this work to divide the network resources between the slices, other situations in the network environment may lead the infrastructure to react to the requirement variations and dynamically adjust its resource distribution. For instance, the initial placement of a fog service may not meet application requirements throughout the vehicle's entire trip due to its mobility. As a result, the fog

service may be migrated to other fog nodes that better meet application requirements based on the new vehicle location. In this context, Figure 5.15 presents the number of service migrations per vehicle performed in each slice. Meanwhile, Figure 5.16 presents the average time required to perform service migrations. In general, the different considered scenarios do not have a great impact on service migration time. However, among the slices, Slice 2, which selected the best links in terms of bandwidth, provides the lowest time to migrate a service. That lower time impacts the overall number of service migrations performed in that slice. As shown in Figure 5.15, indeed, Slice 2 presents the highest number of migrations compared to the two other slices in the same scenario. Another interesting result is that placing the fog services in vehicles may lead the network to perform fewer service migrations. As some works point out, vehicles that run close to each other tend to stay close for some time. This behaviour may keep the fog service running closer to its user for a longer time without the need for migration. As a final consideration on fog service migration, it is worth highlighting that the average times reported in Figure 5.16 are rather long because of the very challenging conditions under which the experiments were conducted. Each scenario indeed simulates the behaviour of 80 vehicles, and many service migrations can occur in parallel, thus sharing the resources of a network slice. More information on fog service migration can be found in [121], where it provides an in-depth analysis of this topic in MobFogSim, conducting migration experiments under various conditions.

In addition to the migration process, latency between the vehicle and its fog service also has a key role. Figure 5.17 illustrates how the slices that take advantage of vehicular fog nodes show lower latency. In general, the hybrid scenario proved the lowest latency. However, the *vehicular_ only* scenario also presents competitive results in terms of latency.

Due to their mobility, vehicles may quickly enter and leave a zone where a slice is built (in these evaluations, the average permanency time in a zone is 4:26 minutes). As a result, a slice may need to be dynamically reconfigured to include/exclude vehicles' resources from the set of available resources. Each reconfiguration incurs an overhead. Figure 5.18 presents the number of slice reconfigurations in each scenario. The *vehicular_ only* and *hybrid* scenarios show equivalent results. Indeed, they both add vehicles' resources that meet their requirements, irrespective of whether or not those resources will be used in future. In those scenarios, on average, around 50 reconfigurations are performed within

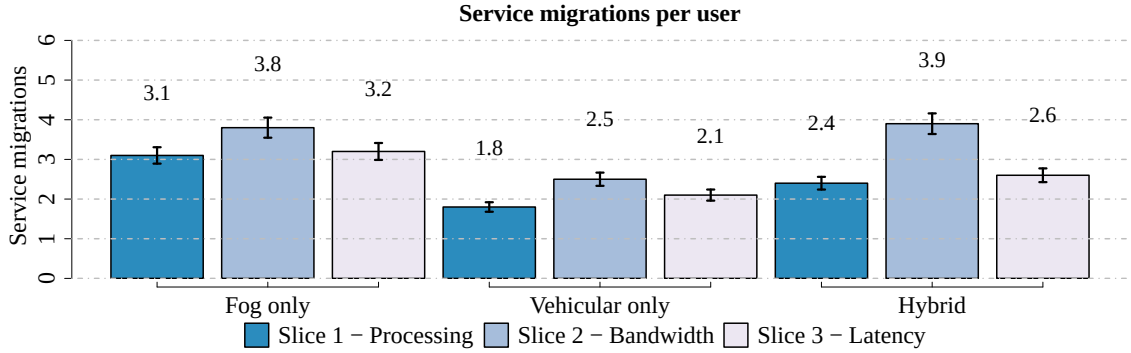


Figure 5.15: Number of migrations performed in each slice.

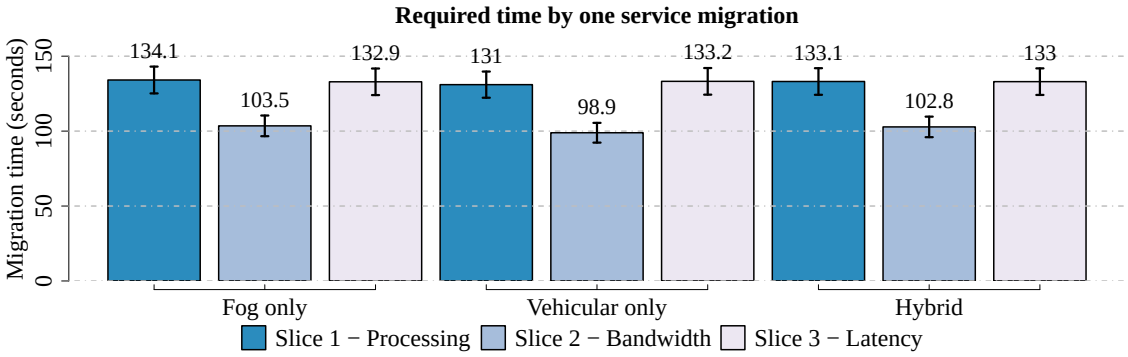


Figure 5.16: Required time to perform service migration.

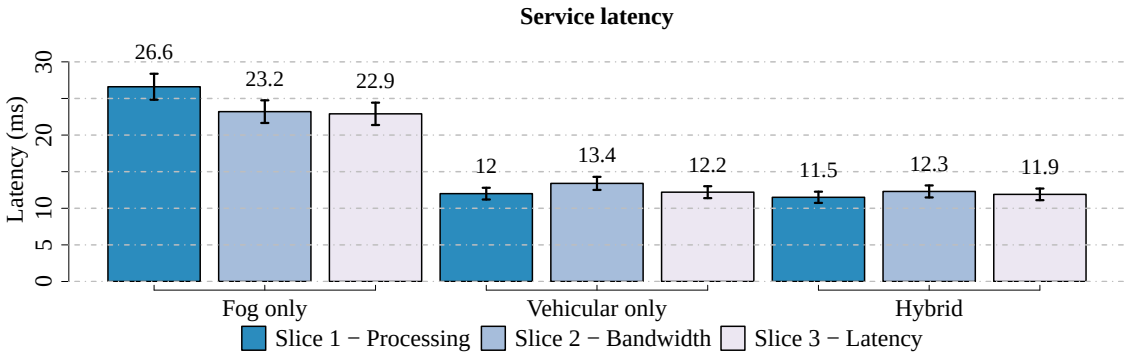


Figure 5.17: Service latency delivered by each slice.

the 60-minute lifetime considered in simulations.

As described in this section, the evaluation of different scenarios (*fog_ only*, *vehicular_ only*, and *hybrid*) reflects some state-of-the-art conclusions. At the same time, it raises new research questions, such as the development of algorithms to get the benefits of the hybrid scenario. Furthermore, these experiments show some of the capabilities of MobFogSim and highlight which scenarios can be simulated with this tool.

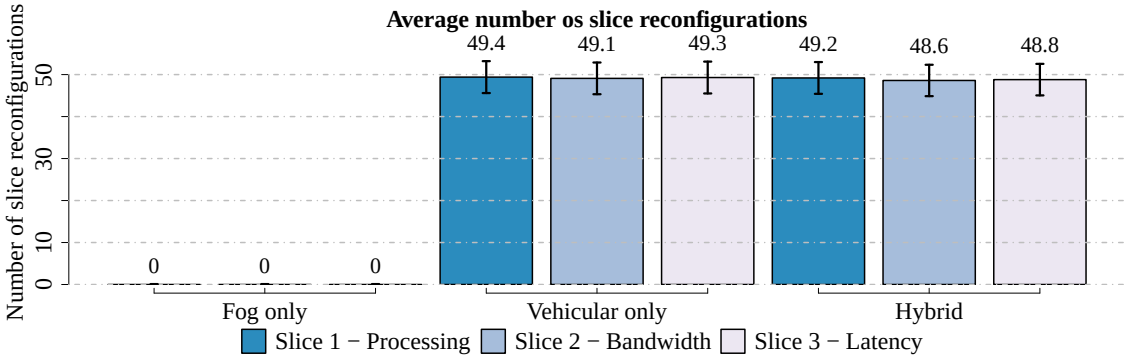


Figure 5.18: Number of slice reconfigurations.

5.4 Chapter Conclusions

5G, 6G, and beyond networks need to accommodate users with very different mobility patterns and application requirements. Network slicing could be important in dealing with such a diverse and heterogeneous environment. Network slicing creates several logical networks on top of the physical one, where each logical network is tailored to address the requirements of a specific group of users and applications. Despite bringing attention from academia and industry, further studies about the performance of network slicing, in special in serving mobile users' requirements, are scarce in the literature.

This chapter presents extensive experiments regarding the role of static and dynamic network slicing in dealing with different mobile user priorities in fog/edge computing environments. Results show that both static and dynamic slicing can introduce traffic priority to deal with different service migration requirements. However, network resources may be underutilised in static slicing in some periods. On the other hand, the capacity of dynamic slicing to scale-up or scale-down provides a better utilisation of network resources. However, dynamic resource reallocation is sensitive to computing overhead, and resource requests may need to wait for reassignments to occur. The overhead of dynamic slicing is further evaluated in the following chapter. In addition, network slicing and fog computing solutions were evaluated in three different VANET scenarios: i) *fog_ only*, wherein fog nodes are deployed only at the edge of the fixed infrastructure; ii) *vehicular_ only*, where fog nodes are provided only by nearby vehicles; and iii) *hybrid*, which combines the previous two. Results show that the *vehicular_ only* approach can reduce the latency provided to end-user devices by around 50% with respect to the *fog_ only* approach. However, the *hybrid* approach performs the best results among those three scenarios.

Chapter 6

Overhead and Performance of Dynamic Network Slice Allocation

Despite the benefits of dynamic network slicing, such as better resource utilisation, avoiding underutilisation and overcapacity scenarios, presented and discussed in previous chapters of this thesis, they come with one additional management burden: the significant increase in the complexity of the operations. In addition, slice modifications are not instantaneous, and resource allocations may need to wait for reassignments to occur. Those impacts of the overhead in slicing reallocations in terms of network performance and the users' QoS are not fully investigated in the literature. In special, the slice orchestration, which is required to achieve such benefits, is not straightforward [19]. In this context, the "main challenge is balancing the coordination overhead with the efficiency of reconfigurations" [45].

In previous chapters of this thesis, a research is presented regarding the impact of network function placement and migration process for user mobility support in fog computing environments [121]. In addition, it is proposed and evaluated the use of dynamic network slicing in serving user mobility support by boosting the service migration process in edge computing environments [66, 67, 68], and vehicular networks [70]. However, for the best of our knowledge, those previous works and the remainder of the state-of-the-art focus on the *benefits of network slice reconfigurations* in scenarios of user mobility support and/or traffic variations.

On the other side, this chapter focuses on the *overhead of the dynamic slicing allocation* process and *how it impacts the network performance*. In particular, this work evaluates the required time (overhead) to proceed with the slice allocation process in 5G/fog computing infrastructures in the context of traffic variations from mobile users and how it impacts the network performance.

The work proposed by [48] presents one systematic review of the state-of-the-art network slicing testbeds for deployment evaluation. However, only two of the 21 cited works from that systematic review report the performance evaluation regarding the required time to perform slice allocations and reconfigurations [59, 62]. Despite the existence of some additional works reporting those values [166, 148], it highlights the scarcity of performance evaluation of network slicing allocation, deployment and reconfiguration in the literature. Based on that, a mechanism that estimates the operation costs and how

it impacts the slice performance is still required. This work presents a mathematical model that describes the expected time to deploy and modify a network slice instance in runtime. The model describes embedding costs required to allocate a functional slice instance, *e.g.*, VM and VNF allocation and boot-up, and reconfiguration costs related to runtime orchestration, *e.g.*, scale-up and scale-down.

Based on values provided by the proposed mathematical model, some experiments present how the reconfiguration overhead on network slicing impacts the network performance. Different network scenarios in 5G/edge computing environments, which include user mobility support, were considered. In this work, the slice reconfiguration is based on dynamic bandwidth demands on the transport network triggered by service migrations. The content of this chapter is published in a peer reviewed journal paper [69]. The contribution of this chapter is twofold.

- A proposed delay allocation model that describes the required time to allocate a slice instance based on its VNFs. That model calculates the total time required to allocate the slice over the infrastructure considering the VNF placement and the VNF isolation;
- An analysis of how the reallocation delay of dynamic slicing impacts the network performance of its users over time. That impact is evaluated in terms of allocated bandwidth, service outage, and service latency.

The remainder of this chapter is organised as follows:

- Section 6.1 introduces the network slicing scenarios considered in this work and the evaluation environment;
- Section 6.2 introduces the dynamic slicing allocation model which is evaluated in Section 6.3;
- Section 6.3 also presents one extensive performance evaluation of the dynamic slicing allocation problem in the context of mobile user support, specifically focusing on allocation delay;
- Section 6.4 presents the conclusions and final remarks of this section.

6.1 Dynamic Slicing Scenarios and its Overhead Impact

Although the ability to select tailored resources to compose an allocation of one network slice instance, the network infrastructure needs to deal with different scenarios of resource changes. Changes in user density due to user mobility, issues with reliability and availability aspects (nodes or links outages), or variable demand intrinsic to services may lead to changes in required resources over time. In particular, this work focuses on demand variations due to user mobility. In such a context, the network infrastructure must continuously serve its users while they move along different locations. Each service accessed

by these users should be placed and, if needed, migrated to the server that provides them with the best conditions. These conditions are based on the service priorities, *e.g.*, lowest latency or highest reliability.

In the context of NS allocation, the network resources can be assigned to each slice either statically or dynamically. A static network slice receives a fixed portion of network resources according to its demand. This number of resources remains allocated to that slice until the slice is active. On the contrary, dynamic network slicing allows a slice to acquire and release resources according to dynamic demand, thus enabling scaling-up and scaling-down [54]. Further details were discussed in Section 2.2.2.

The reconfiguration frequency is intrinsically related to traffic variations and the slice reconfiguration overhead. That overhead can be described in terms of (1) service outage (allocation delay) and (2) reconfiguration cost [52, 160]. The (1) reconfiguration delay is about the time required for transit from one slice configuration state to the new one. That time comes from deploying new rules for programming switches, which is not instantaneous. In such a scenario, the reconfigurability of the data plane "can lead to some downtime, state loss, and service interruptions for all NFs deployed on the switch and all flows processed by it" [155]. In special, "changing data plane behaviour at runtime without disrupting packet processing remains an open problem" [108]. The (2) reconfiguration cost could be described as the amount of computing, storage and network resources requested or released from one slice configuration state to another one. In some works, the *cost* of resource management operations is described in terms of monetary costs [5, 28, 119]. However, that metric indirectly uses computing costs, *i.e.*, computing, storage and network usage, as an input. Monetary costs are out of the scope of this work.

Low reconfiguration overhead could enable the network to perform frequent resource reallocations. Frequent reallocations optimise resource utilisation by minimising overcapacity and underutilisation scenarios. However, "while this increases resource efficiency, it also leads to instability" [155]. On the other hand, network infrastructures that demand higher computing overhead to perform their slice reconfiguration may end in less frequent reconfigurations.

Despite the development of network slicing solutions in recent years, some challenges still are open, especially for mobile user scenarios. One performance evaluation of some of these scenarios is present in this work. In particular, the scope of the dynamic slicing scenarios considered in this work is discussed below:

- *Slicing reallocation overhead* - Once a slice is created and resources are allocated to it, there are computational and time costs to resize it. This cost may depend on the physical and logical network characteristics. Network appliances from different brands and software that manage it present different efficiency rates to reallocate slices. The delay of this reallocation process may impact the network performance. Section 6.2 introduces one allocation delay model that describes the required time to allocate one slice instance. In this work, the slicing reallocation overhead is measured by the required time (delay) to reconfigure the slice instance;

- *Slicing reallocation frequency* - the frequency at which reallocation happens. That frequency is strictly related to the reallocation cost/delay: the lower the cost/delay,

the higher the reallocation frequency can be. Moreover, the frequency could be adapted to the priority of slices. For instance, high-priority slices could free their resources with a lower reallocation frequency than low-priority ones. This way, high-priority slices would incur a lower reallocation cost when they receive their resources back. The impact of Slicing reallocation frequency is evaluated in Section 6.3.

In this work, the *dynamic slicing allocation problem* is investigated focusing on the reallocation overhead in a dynamic user mobility scenario and how the service outage impacts the network performance. Different factors can result in an increasing traffic variation. However, this work assumes a service migration between two different physical nodes as a significant event in the evaluation scenario. In the assumed scenario, the network infrastructure adopts the follow-me cloud concept. Then, some services may be migrated from one source fog node to another to serve its mobile users as close as possible. In this scenario, a network slice can request a resource reallocation to scale-up its link bandwidth to proceed with the migration process. Figure 6.1 illustrates that scenario in which a bandwidth slicing reallocation is triggered by a service migration. In that figure, the physical infrastructure has a 800Mbps link between its nodes. Initially, 100Mbps is allocated for one slice instance. However, once a higher demand, *e.g.*, one service migration, is identify by the Management and Orchestration System (MANO), the bandwidth of the link between the source and destination node of the service migration is scaled-up from 100Mbps to 400Mbps.

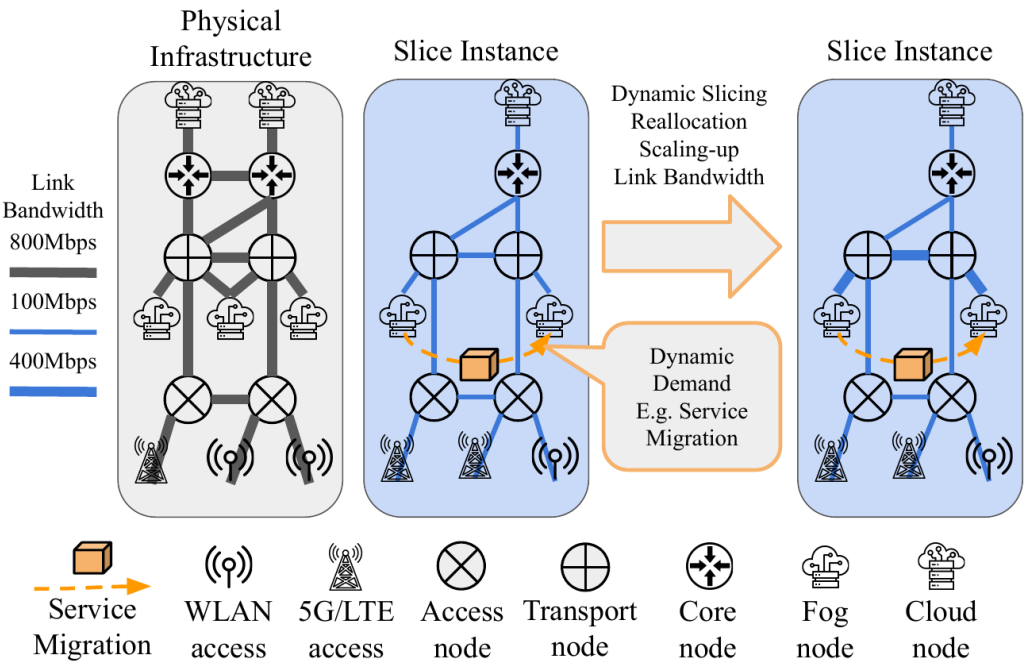


Figure 6.1: Example of dynamic slicing with bandwidth reallocation. Mobile users trigger traffic variations in the network, *e.g.*, service migration. Then, the slice requires a resource reallocation to scale-up its link bandwidth in the transport network.

6.2 Allocation Delay Model

Despite the relevance of network slicing placement algorithms, the performance of NFV Management and Orchestration (MANO) systems also has a relevant aspect in the performance of network slicing. In the face of the relevance of that topic and the lack of models that describe and calculate the required time to deploy a network slice instance in literature, this work proposes a model to fill that gap. This section presents a mathematical model that calculates the expected time to deploy a network slice instance.

Benchmarking different MANO systems can lead the infrastructure providers to choose the best system to orchestrate their virtual infrastructure, based on, *e.g.*, functional KPIs like monitoring systems support, and operational KPIs like slicing allocation delay. Estimating the time required to instantiate the network slicing is critical to modelling dynamic network slice algorithms. That time to allocate and reconfigure a slice instance will impact the network slicing reconfiguration frequency. The proposed model considers some benchmark results that describe the performance of some VNF MANO systems.

Section 6.2.1 presents the definitions of Key Performance Indicators (KPIs) for MANO system performance evaluation. Based on that, Section 6.2.2 presents the proposed model for the slice allocation delay.

6.2.1 KPI for MANO Systems Performance Evaluation

The performance of the MANO systems in allocating and managing the VNFs that compose a slice is crucial to defining the ability of the infrastructure to provide dynamic slicing support properly. However, the scarcity of MANO performance benchmark reports and a standard KPI definition for these systems makes evaluating such systems challenging.

Yilma *et.at.*[166] propose classifying the MANO systems performance KPIs into two categories. The first one, the Functional KPIs, describes non-run-time characteristics, like the maximum number of VNFs supported by the MANO system, DevOps support, or support to specific monitoring systems. The second category proposed in [166], named as operational KPIs, describes runtime characteristics and is closely related to the slice deployment process. In particular, this work focuses on operational KPIs. In operational KPIs, the delay required to instantiate a slice is based on three operations: the Onboarding Process Delay (OPD), the Deployment Process Delay (DPD), and the Runtime Orchestration Delay (ROD). The delay allocation model proposed in this work is based on an adapted version of the slicing operations lifecycle defined by [166]. Figure 6.2 illustrates the slicing deployment steps adapted for the scenario assumed in this work.

In this work, ODP and DPD are assumed as part of the virtual network embedding process meanwhile ROD is related to the slice reconfiguration process. In the slice allocation process, the first phase (ODP) starts with preparing the environment, which includes the allocation and boot-up of the VM that will host the VNF. The second phase starts with the allocation of the VNF in the DPD phase. The third phase (ROD), the dynamic orchestration of the slice, implements operations to scale or finish the slice instance.

In detail, the Onboarding Process Delay is the time required to prepare the environment to host the VNF. In this phase, a VNF image is created and sent to the node that

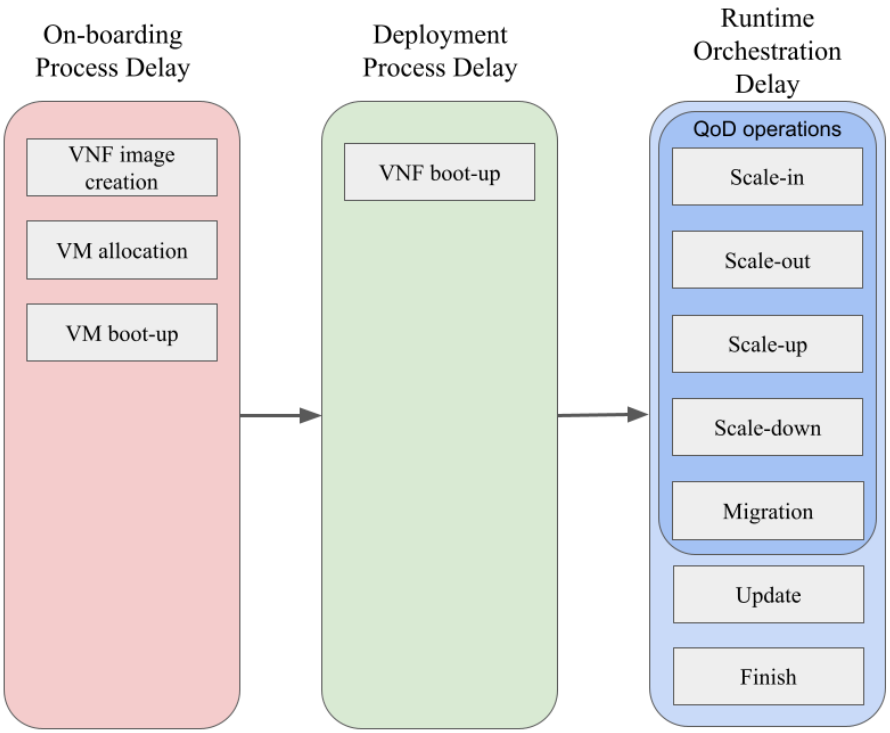


Figure 6.2: Slicing Operations lifecycle. Adapted from [166].

will host that VNF. The VNF image contains the VNF service and related files describing the VNF operations. Those files contain the VNF descriptor that indicates the network and resource requirements, the NS descriptor that indicates the required QoS and QoE levels of that slice instance, and the VNF Forwarding Graph that indicates the order in which the VNFs will be chained. Based on the requirements described by the VNF image, a VM that offers the VNF's required resources should be prepared to host that service. After the allocation and boot-up of the VM, the environment will be ready to host the VNF service. The allocation and boot-up of the VM can be made once a new VNF is defined to run on that infrastructure node or is previously allocated with pre-defined resources. In the second option, the delay in instantiating a VM is reduced. However, the characteristics of the VM in terms of resources may not be thoroughly adjusted to serve the VNF requirements.

Once the VM is ready to host the VNF and the VNF software is already inside the VM, the second phase can start. In this phase, the software that describes the VNF is booted up. The VNF will allocate the resources from the VM described in the VNF descriptor and start linking that VNF to other VNFs that compose its chain, as described in the NS descriptors and the VNF Forwarding Graph. The time elapsed in this phase is defined as DPD in the model.

The Runtime Orchestration's last phase starts after the slice instance is operational. The runtime orchestration operations are requested to adjust the slice configuration in terms of resources dynamically, *e.g.*, scale-in and scale-up; placement, *e.g.*, VNF migration; or the slice settings, *e.g.*, update or finish a VNF instance. The operations strictly related to performance metrics pointed as Quality of Decision (QoD) operations in Figure 6.2 can change the availability of the slice's resources. In a scenario with dynamic

requirements, *e.g.*, with user mobility, those operations have a significant importance in the dynamic slicing context. The performance of those operations will define the required time to reconfigure a slice instance, which impacts, among others, the service outage discussed in this work. Based on the Onboarding Process and Deployment Process operations, as illustrated in Figure 6.2, a slice allocation model is proposed to describe the required time to instantiate a slice. The model is described in Section 6.2.2.

6.2.2 Slice Allocation Model

This work proposes a slice allocation model that estimates the required time to allocate a slice instance based on its VNFs. The proposed model is based on Operational Key Performance Indicators for network slicing MANO systems defined by [166]. The work present in [148], [104], and [62] report network slicing benchmarks in testbed infrastructures that include VMs and VNFs allocation. The proposed model considers the performance evaluation of those experiments in its definition.

Assuming the model, it can estimate the time needed to prepare the environment, allocate the required resources and boot-up one functional slice instance. In this proposed model, the network slicing deployment time (NSDT) is defined as the Onboarding Process Delay (OPD) + Deployment Process Delay (DPD). In this work, the Runtime Orchestration Delay (ROD) is not explicitly described in the model. However, without losing generality, the operations pointed as Quality of Decision (QoD) operations in the runtime orchestration process, as seen in Figure 6.2, such as scale-in, scale-up, scale-out, scale-down, and VNF migration are variations of VMs and VNFs allocation and boot-up operations described in the slice onboarding (OPD) and deployment (DPD) process. In that context, the proposed model can be used to estimate the reconfiguration time of an active slice. Equation 6.1 summarises the model, which is composed of the sum of OPD and DPD times. Table 6.1 describes the parameters that compose the NSDP model. The F set defines the VNFs that compose the network slice instance, while set N defines the infrastructure nodes that can host those VNFs.

$$NSDT = OPD + DPD \tag{6.1}$$

The proposed model describes the expected allocation and deployment time for one slice instance. The model considers that a slice is composed of a chain of VNFs and an ordered set of links that connect each pair of VNFs. A virtual machine on an infrastructure node hosts each VNF. One slice can have multiple VNFs on the same physical node, and a physical node can host multiple VNFs from one or more slices. The model assumes that, if a slice allocation includes more than one VNF running on the same physical node, those VNFs are deployed sequentially. VNFs assigned in different physical nodes are allocated in parallel. In addition, a VNF can be dedicated to a slice instance or shared among different slices. In that scenario, the allocation delay and every other allocation cost related to that VNF is assigned to the first network slice that requested that VNF. Figure 6.3 illustrates one example of network slice allocation.

In the proposed example, Slice Instance 1 is composed of 4 VNFs. The network is composed of four nodes. Five links and two virtual links connect the nodes and the

Table 6.1: Description of the mathematical symbols of the Network Slicing Deployment Time Model

Symbol	Description
$N = \{n_1, n_2, n_3, ..., n_k\}$	Set of infrastructure nodes, where $k = \|N\|$
$L = \{l_1, l_2, l_3, ..., l_j\}$	Set of infrastructure links, where $j = \|L\|$
$V = \{v_1, v_2, v_3, ..., v_i\}$	Set of VM $v \in V$ in $n \in N$, where $i = \|V\|$
$F = \{f_1, f_2, f_3, ..., f_x\}$	Ordered set of VNFs from a slice instance, where $x = \|F\|$
$E = \{e_1, e_2, e_3, ..., e_z\}$	Ordered set of links that connect each ordered pair of VNFs $f \in F$, where $z = \|E\|$
$G_n = \{f_1, f_2, f_3, ..., f_g\}$	Subset of F, *i.e.*, $(G \subseteq F)$, which contains the VNFs from the slice instance that run on node $n \in N$, where $g = \|G\|$
$H = \{n_1, n_2, n_3, ..., n_h\}$	Subset of N, *i.e.*, $(H \subseteq N)$, which contains the nodes that run at least one VNF $f \in F$, where $h = \|H\|$
τ_f	Creation time of the VNF image $f \in F$
$\eta_{v,n}$	Allocation time of VM $v \in V$ in $n \in N$
$\nu_{v,n}$	Boot-up time of VM $v \in V$ in $n \in N$
$\mu_{f,n}$	Boot-up time of VNF $f \in F$ in $n \in N$
σ_f	Boolean value which indicates whether VNF $f \in F$ has already instantiated by other slice

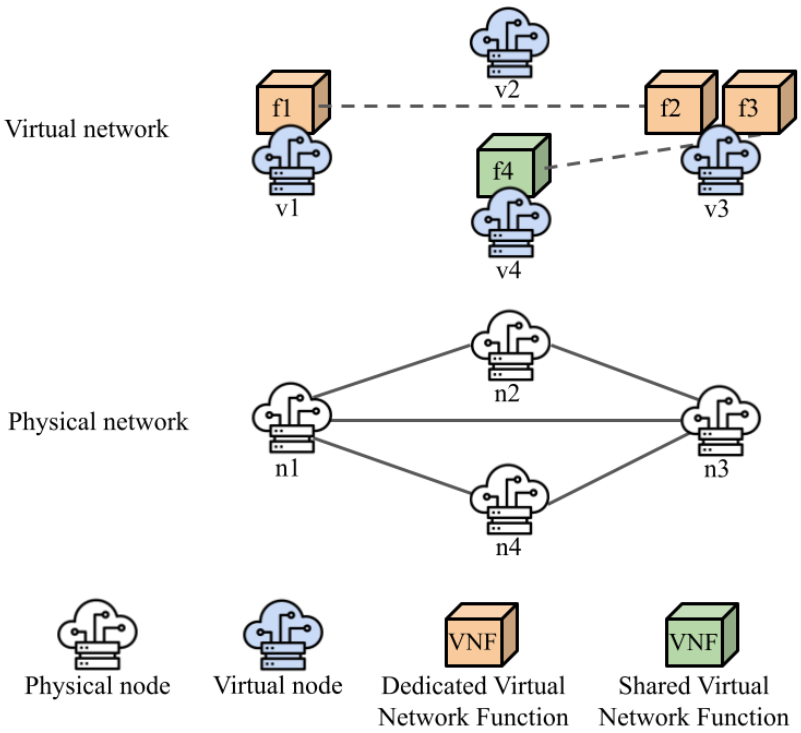


Figure 6.3: Example of service function chain in a network slice.

virtual machines. In that example, Slice Instance 1 is composed of the VNFs f_1, f_2, f_3, and f_4, which run on nodes n_1, n_3, and n_4, respectively. In that example, some topics can be highlighted: 1) in the network slicing process, a slice instance, *i.e.*, the logical network, does not need to select all available physical nodes, *e.g.*, n_2; 2) multiple VNFs can be assigned in a single physical node, *e.g.*, VNFs f_2 and f_3 run on the same node(n_3) and will

be allocated sequentially in that node. In some scenarios, multiple VNFs are allocated in the same node for performance purposes, *e.g.* due to high traffic and low latency requirements between them. The number of VNFs allocated in a physical node depends on the available resources provided by that node; 3) different network slice instances can share one or more VNFs, *e.g.*, VNF f_4. In that example, VNF f_4 is already allocated and currently operated by another slice, *e.g.*, Slice Instance 2. The variable $\sigma_{f_4} = T$ defines that scenario. Both Slice Instance 1 and Slice Instance 2 will share the VNF f_4. In this case, f_4 will not consume resources to be allocated, nor it will impact the allocation of Slice Instance 1. The remainder of the VNFs are dedicated to serving only one slice instance.

Based on that described scenario, the model can estimate the required time in the network slice embedding process. That process, which is composed of two phases, OPD and DPD, is illustrated in Figure 6.2. In this proposed model, the Onboarding Process Delay (OPD) is composed of Equations 6.2 and 6.3; meanwhile, the Deployment Process Delay (DPD) is composed of Equations 6.4 and 6.5.

The Onboarding Process Delay presents the expected time the infrastructure requires to prepare the VM environment. That step includes downloading the VNF image, which describes its settings and resource requirements, and the allocation and boot of the VM in the node. This step in the model calculates the time when the last VM is fully booted up, which finishes the Onboarding Process and enables the infrastructure to be ready for the Deployment Process. This step is composed of Equations 6.2 and 6.3. Table 6.1 summarises the symbols and parameters of the proposed model.

This work defines Equation 6.2 as the function $T_1(n)$. Given N as the set of infrastructure nodes and H as a subset of N ($H \subseteq N$) of nodes that hosts at least one VNF from the slice instance. Equation 6.2 calculates the required time for one node $n \in H$ to finish the onboarding process for all its VNFs. In this equation, for each VNF from one slice instance that runs on node n ($f \in G_n$), it is computed three variables that compose this step: required time to create/download the VNF image (τ_f), the allocation time of VM v on the node n ($\eta_{v,n}$), and the boot-up time of VM v on the node n ($\nu_{v,n}$). The summation of those variables describes the onboarding process delay for node n. Once multiple VNFs from the same slice instance can run in the same physical node, this equation calculates the required time to allocate and prepare the VMs in node n sequentially. VNFs running on different nodes can be allocated in parallel. In addition, a VNF already allocated and currently in operation by another slice, defined in the model as $\sigma_f = TRUE$, does not consume resources from this current slice allocation.

$$T_1(n) = \sum_{\forall f \in G_n} \begin{cases} \tau_f + \eta_{v,n} + \nu_{v,n}, & \text{if } \sigma_f = FALSE \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

Once Equation 6.2 calculates the onboarding process delay for each node $n \in H$, Equation 6.3 finds the last node to be ready for the next step. In this equation, the max function calculates the highest value among all nodes that host at least one VNF from the slice instance ($\forall n \in H$). As described above, $T_1(n)$, *i.e.*, Equation 6.2, calculates the

onboarding process delay for each node n. Then, Equation 6.3 calculates the onboarding process delay for the slice instance considering all its VNFs.

$$OPD = \max_{\forall n \in H} T_1(n) \tag{6.3}$$

After the Onboarding Process is completed, *i.e.*, the VNF image is copied into a node and the VM is operational, then it is possible to proceed with the Deployment Process. In this proposed model, the DPD is composed of Equations 6.4 and 6.5. Those equations describe the required time to boot-up the VNF, which includes the time to make the VNF fully operational, *i.e.*, starting the VNF boot within the prepared VM.

This step in the model calculates, among all the nodes n that compose the slice, the time when the last VNF f is operational. This work defines Equation 6.4 as $T_1(n)$. Similarly to Equation 6.2, given H a subset of N ($H \subseteq N$) of nodes that hosts at least one VNF from the slice instance, Equation 6.4 calculates the required time for one node $n \in H$ to finish the deployment process for all its VNFs. The $\mu_{f,n}$ variable defines the required time to boot-up the VNF f into node n. σ_f is a boolean variable that defines whether VNF f has instantiated by other slice already. In this case, that VNF does not impact the deployment process of that slice instance. The summation calculates that value for each node once it is assumed that its VNFs are sequentially booted up.

$$T_2(n) = \sum_{\forall f \in G_n} \begin{cases} \mu_{f,n}, & \text{if } \sigma_f = FALSE \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

Finally, Equation 6.5 finds the highest value required by all nodes to finish the deployment process. Those values are calculated by Function $T_2(n)$ in Equation 6.4 for each node n.

$$DPD = \max_{\forall n \in H} T_2(n) \tag{6.5}$$

Many operations in the runtime orchestration process, as seen in Figure 6.2, are variations of onboarding and deployment operations. Operations such as scale-in, scale-up, scale-out, scale-down, and VNF migration consist of allocating or freeing resources for VMs and VNFs. In that context, without losing generality, the proposed model can be used to estimate the reconfiguration time of an active slice given the set of VNFs and physical nodes and links.

6.3 Performance Evaluation of Dynamic Slicing

This section presents the performance evaluation of network slice reconfiguration. The experiments evaluate the impact of dynamic resource reallocations of slices over time. The demand variations due to user mobility are the focus of this work. So, this section

analyses the dynamic resource allocation in the context of traffic variation triggered by service migrations.

Similarly to previous scenarios discussed in this thesis, mobile users access services present in the infrastructure. The scenario is based on the follow-me cloud concept, in which the services may be migrated from one fog node to another that best fits the users' requirements, especially in terms of latency. Once the services migrate among the fog nodes, the migration triggers demand variation on the network infrastructure. In this context, the slices can request additional bandwidth to improve the service migration process. Then, the network slices may require one slice reconfiguration to scaling-up its link bandwidth to serve that service migration. That process is illustrated in Figure 6.1.

In this experiment, the slice's idle resources can be assigned to other slices with higher demand. Different values for that delay related to the reallocation process, which is characterised in terms of the required time to perform such reallocations, are considered in this work. In this experiment, two research questions are considered: (1) How do different values of the reallocation delay impact the slicing reallocation frequency? and (2) How does that reallocation frequency impact the performance of the service migrations? The evaluation of (1) is made in terms of reallocation frequency, the amount of bandwidth granted by each slice request and the current allocated bandwidth per slice over time. As a result of (1), the evaluation of (2) is made in terms of the number of migrations and service delays.

Realistic user mobility was used to evaluate the impact of mobile users in the slicing resource management. In addition, simulations' network settings are defined based on testbed data. Section 6.3.1 summarises the simulation setup adopted in this work. Section 6.3.2 presents a performance evaluation of bandwidth slicing reallocation over time to serve service migrations.

6.3.1 Simulation setup

In this work, the evaluated scenarios are built in the MobFogSim simulator. As one of the main focuses of this work, the user mobility adopted in the simulations is based on a dataset of realistic user mobility patterns. The mobility of each user is based on vehicles from Luxembourg provided by the LustScenario database [31]. The simulated scenarios selected 2400 different mobility patterns from that dataset.

The values used as parameters for fog resources were based on real infrastructure [121] equivalent to the evaluated scenario. Further details were described in Section 4.7. The infrastructure used in this evaluation is composed of 16 fog nodes distributed uniformly on a 5 km x 5 km map. 64 access points provide a wireless connection between users and the infrastructure. The connection between the fog nodes is made by one 74 Mbps link with 3.47 ms latency. In the service placement, the fog node that offers the lowest latency for the user is the one selected to host him/her service. These services/applications are running on a virtual machine. In the case of service migrations, each virtual machine has 128 MB of RAM to be migrated. Table 4.1 summarises the values used as parameters to define the resource availability and demand of resources used in this work. The computing power of services and fog nodes is presented in MIPS (Millions of Instructions Per Second).

Table 6.2: List of input parameters and their values based on the testbed experiments of [121] and user mobility dataset [31]

Parameters	**Value**
Client processing requirement	2901 MIPS
Client storage requirement	4 MB
Server storage requirement	412 MB
Client processing capacity	46534 MIPS
Server processing capacity	3234 MIPS
Average access point delay	4.78 ms
Average delay between servers	3.47 ms
Client RAM requirement	49 MB
Server RAM requirement	128 MB
Average access point bandwidth	13640 Kbps
Average bandwidth between servers	74148 Kbps
Size of container execution state during live migration	12.8 MB
Access point coverage (radius)	500 m
Number of fog nodes	16
Density of fog nodes per access points	1:4
Migration strategy	Lowest latency
Migration point policy	Static (40 meters)
Average user speeds	22.3 Kmph
Number of user mobility patterns	2400
Number of users per simulation	80

In this experiment, the users are assigned to one of two slices placed on the infrastructure. The slices share the same "physical" infrastructure and have the same topology, *i.e.*, the same set of nodes and links. At the beginning, each slice received 50% of the available bandwidth. However, in this experiment, the traffic distributions, *i.e.* the number of users accessing the slices' resources, vary in each simulation scenario. In this experiment, the bandwidth reallocation occurs in the transport network.

The simulation loads 80 users distributed into two slices, Slice 1 and Slice 2. The number of users assigned to each slice sets the traffic distribution of the slices. Three different scenarios of traffic distribution were evaluated in this experiment. In Scenario (1), Slice 1 received 10% of the total number of users loaded in the simulation, while Slice 2 received 90%. In Scenario (2), Slice 1 received 20% of users while Slice 2 received 80%. Finally, in Scenario (3), Slice 1 serves 40% of the users, and Slice 2 serves the remaining 60%. The level of users distribution stays stable throughout the entire simulation. The simulation ends if all users finish their routes or the simulation time hits 60 minutes. The delay assumed in the slice's reconfiguration was based on benchmarks found in the literature [148, 166, 59, 48, 62]. The delay to deploy a slice reconfiguration reported in the literature varies from 131 seconds [148], 90 seconds [62], and 66 seconds [166]. The network slice allocation in infrastructures geographically distributed across different countries and continents might present different values, *e.g.*, 578 seconds [30]. That scale of geographical distribution is out of the scope of this work. Based on values from [148, 62,

166] and assuming the tendency to decrease those values over time due to the performance improvements in Management and Orchestration Systems, the range of values considered for the reconfiguration delay varies from 2 to 60 seconds. The values 2, 30, and 60 assumed as reconfiguration delays for the slice operations are fixed during the simulation. Table 6.3 summarises the parameters of the simulated scenarios.

Table 6.3: Characterisation of the simulated scenario.

Parameters	**Value**
Number of slices	2
Initial bandwidth distribution per slice	50%
Traffic distribution per slice (% of total users)	10%, 20%, and 40% in Slice 1, and 90%, 80%, and 60% in Slice 2
Slice's reconfiguration delay	2, 30, and 60 seconds

6.3.2 Results

The dynamic bandwidth slicing reallocation was evaluated based on smart city scenarios and the simulation setup described above. This section presents the results of these experiments. One of the main advantages of dynamically (re) allocating resources in the slicing context is the possibility of redistributing resources to entities based on those demands. In that scenario, idle resources, *i.e.*, bandwidth, could be better used in slices with higher demand. The experiments introduced in this section present the impact of different delays in the reconfiguration process. The results are presented considering a 95% confidence interval based on the average of 30 simulation runs.

Figure 6.4 presents the number of reconfiguration requests by each slice granted by the infrastructure. It is assumed that each slice in the simulation received 50% of the physical resources at the beginning of the simulation. Despite the equal resource distribution, different workload levels were assigned to the slice. The results are grouped by the different values of reconfiguration delay adopted in the simulation. Figure 6.4 (a) presents the number of reconfigurations considering 2 seconds between the reconfiguration being requested by the slice and the reallocation being completed. Figure 6.4 (b) shows the results considering a reconfiguration delay of 30 seconds. Finally, Figure 6.4 (c) assumes 60 seconds as the reconfiguration delay value. The results presented in Figure 6.4 show the number of attended reconfiguration requests based on the different reconfiguration delays.

The main focus of this experiment is to evaluate the impact of different reconfiguration delays on the number of reconfigurations attended by the infrastructure. The results show that the lower the reconfiguration delay, the higher the number of reconfigurations the infrastructure can provide along the slice life cycle. The number of reconfigurations can differ by up to 40% if compared to 2 and 60 seconds as values for the reconfiguration delays. The simulations point out that, considering a reconfiguration delay of 2 seconds, Slice 2 with 90% of the users had, on average, 24.5 accepted requests in the first 10 minutes of simulations. In the same scenario, but considering 60 seconds as reconfiguration delay,

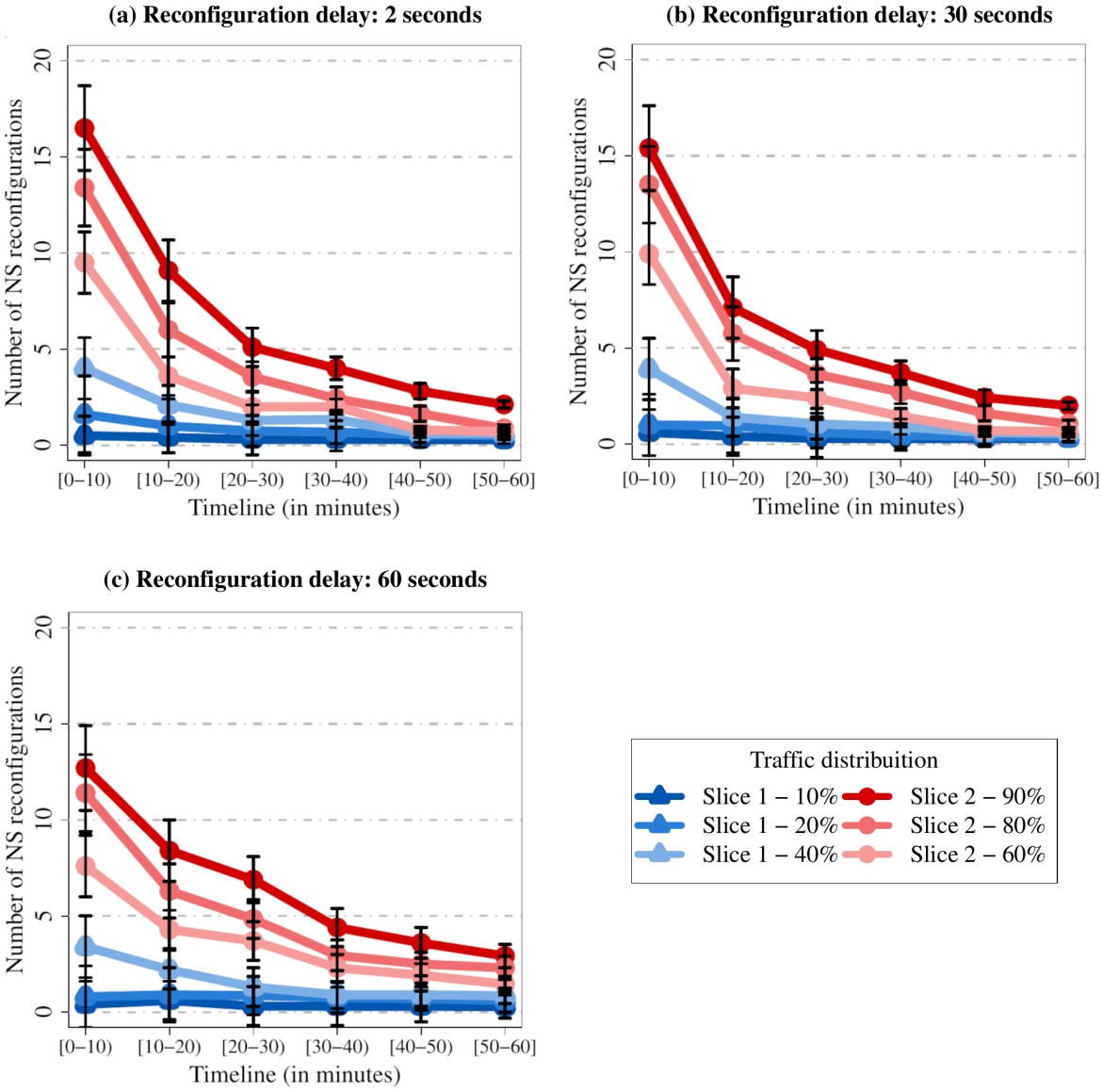


Figure 6.4: Timeline of bandwidth reallocations grouped by reallocation delay.

that number falls 49%, going to 12.7 at the same period in the simulation. These results show that every evaluated workload scenario presents a reduction in reconfigurations as the reconfiguration delay increases.

Furthermore, the number of reconfigurations tends to fall and be more stable as the timeline of the slice lifecycle goes to its end in a scenario with the lowest reconfiguration delay. For example, Slice 1, with 10% of the total users, decreases its reconfiguration requests by almost 60% from the first 10 minutes of the simulation to its last 10 minutes, considering 2 seconds as reconfiguration delay. The same scenario, assuming 60 seconds as reconfiguration delay, presents a reduction of 30%. Slices with a higher workload tend to present a more drastic reduction in the reconfiguration requests. For example, Slice 2, with 90% of users, decreases its requests by 93%, considering 2 seconds as the reconfiguration delay, while that reduction is about 87% assuming 60 seconds. In the last 10 minutes of the simulation, Slice 2 with 90% of users in Figure 6.4 (a) presents 1.8

accepted requests. That number increases as the reconfiguration delay increases until it presents 2.92 requests in Figure 6.4 (c). In general, slices with lower reconfiguration delay present more reconfiguration requests at the beginning of their life cycle than the higher ones. On the other hand, the last 10 minutes of the slices with higher reconfiguration delays tend to present more requests than the lower ones. Those results may indicate that a lower reconfiguration delay enables the slices to get the required resources faster than the higher delay slices. As the slices received the required resources, they do not need to frequently request reconfigurations anymore.

Despite the reconfigurations enabling the network to dynamically balance the resource distributions over time, those reconfigurations may lead the slice to an outage state while that process occurs. As described in Section 6.3.1, the assumed deployment delay in this experiment varies from 2 to 60 seconds for each reallocation request. The total outage time for each slice in its life cycle results from the number of reconfigurations that slice proceeds in that interval, as presented in Figure 6.4 and the time required to execute each reallocation. The results regarding that topic are introduced in Figure 6.5, which presents the average slice outage over 10 minutes due to the slice reconfiguration process. Although the reallocation delay does not significantly impact the number of reconfigurations, as seen in Figure 6.4, the reallocation delay deeply impacts the average slice outage, as presented in Figure 6.5. Low reallocation delay, *e.g.*, 2 seconds, as presented in Figure 6.5 (a), leads the slice to less than 1% of an outage. On the other hand, the higher reconfiguration delay, *e.g.*, 60 seconds, as presented in Figure 6.5 (c), presents 31% of outages in its life cycle.

The number of reconfigurations describes the frequency in which the slices receive more resources. However, the amount of bandwidth each slice received over time is also a significant metric. For that purpose, Figure 6.6 presents the amount of allocated bandwidth received by the slices along their life cycle. Similar to the trends in the reconfiguration frequency, in general, slices with lower reconfiguration delays tend to receive more bandwidth at the beginning of their life cycle than those with higher delays. Furthermore, slices with higher reconfiguration delays tend to receive more resources until their life cycle ends. This may be justified because low-delay slices receive their required resources more quickly than high-delay ones.

Figure 6.7 presents the amount of bandwidth received by the slices in terms of the percentage of physical resources. The figure shows the timeline of the allocated bandwidth of each slice. Initially, each slice starts with 50% of the bandwidth available in the physical infrastructure. The results show that slicing reconfigurations enable the slices to adjust their resources to serve their demand dynamically. In general, the slices tend to receive resources relative to their workload level. For example, slices serving 90% of the users tend to receive close to 90% of the resources, while slices serving 10% of the users receive close to 10% of the resources. As presented in Figure 6.4 and Figure 6.6, slices with lower reconfiguration delays tend to receive their required resources more quickly than the higher ones. It can be observed that slices in Figure 6.7 (a) tend to be more stable than the slices in Figure 6.7 (b), and Figure 6.7 (c).

Based on the requirements of the slice's users, the (re)distribution of resources, especially the slice's bandwidth, significantly impacts users' service performance. As this work

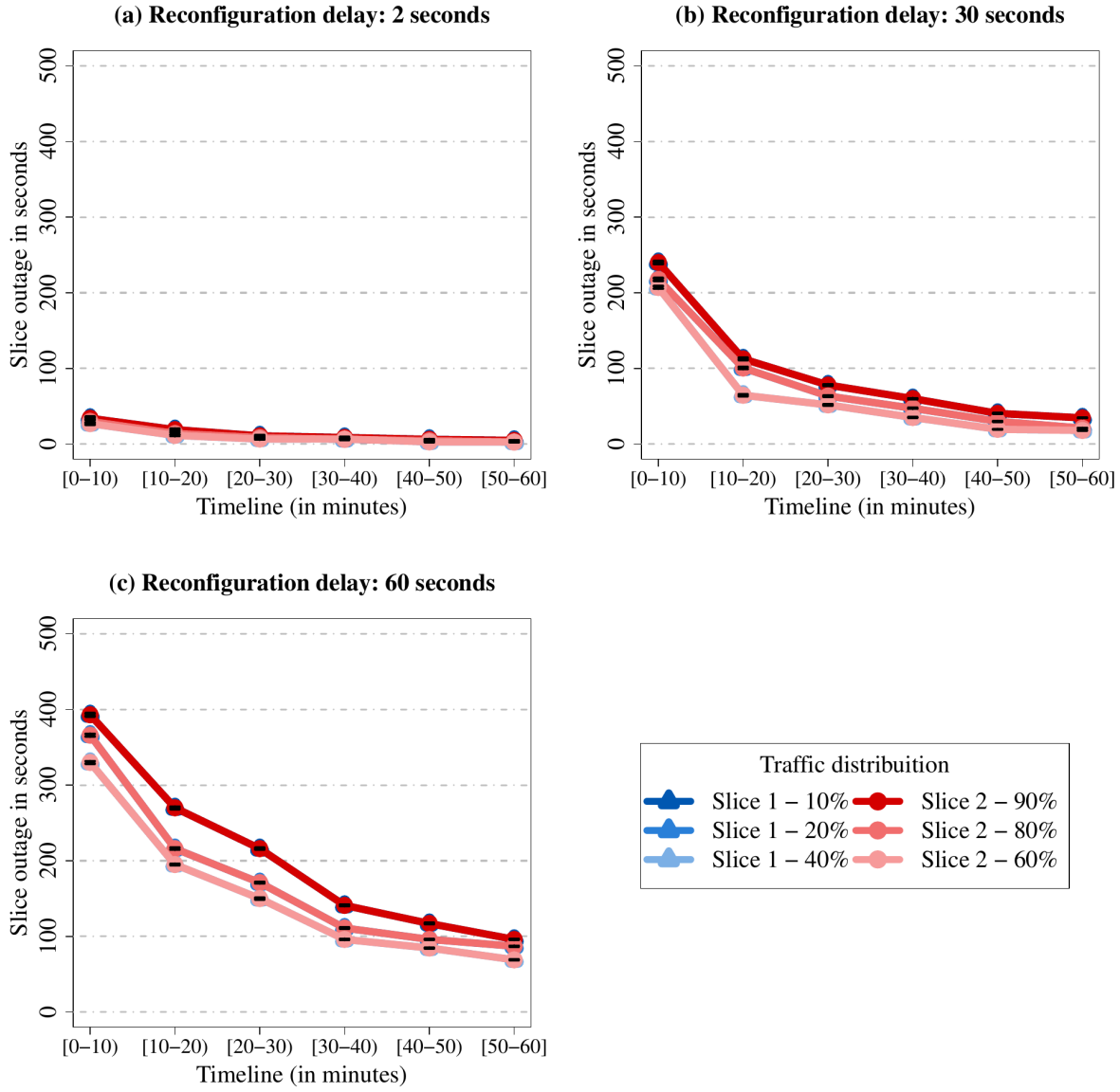


Figure 6.5: Average slice outage over 10-minute intervals.

focuses on evaluating the impact of bandwidth distribution on service requirements, this section includes an evaluation of the impact on the QoS. The resource availability presents some implications for the slice's performance. The performance of service migration and service latency are presented based on the previous scenarios and the results presented above. Figure 6.8 presents the implications of the dynamic bandwidth reallocation on the service migration process. The presented scenarios follow the same settings presented in the previous experiments. The trigger that starts a service migration is based on placing the service in a fog node that provides the lowest latency for the service user. That migration process is described in Section 4.2 and Section 4.4. Algorithm 2 summarises the integration of the slice reallocation and the migration processes.

As identified in the previous results, slices with low reconfiguration delays tend to quickly converge their resources to a stable level that is equivalent to their users' demand. The reallocation delay impacts the speed in which the slices receive the proper amount of

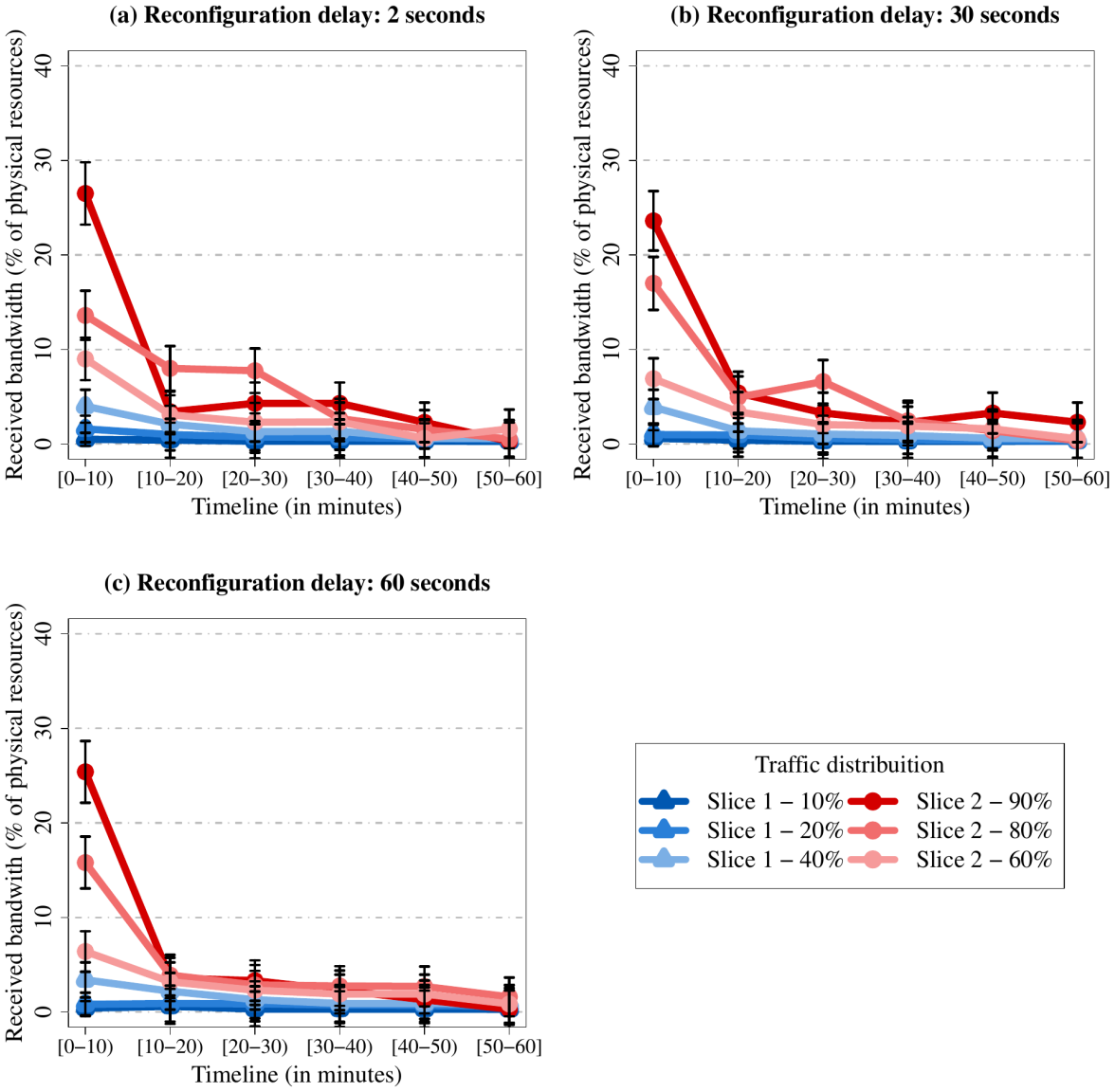


Figure 6.6: Timeline of the amount of allocated bandwidth received by the slices.

resources along with their life cycle, it impacts the load balance and may lead the slices to present resource overutilisation and underutilisation. It may impact the execution of services. In slices with low reconfiguration delay, as presented in Figure 6.8 (a), the average duration of service migrations quickly converges to 116 seconds for all slices. That tendency is slower in the remanding scenarios. In the scenario with 30 seconds or more as reconfiguration delay (Figure 6.8 (b) and (c), the slices do not even converge their resources to a stable level. It impacts the duration of their migrations. In those scenarios, after 60 minutes of simulations, the performance of the service migrations in their slices is still not balanced. In such scenarios, some slices perform service migrations better than the average performance, *e.g.*, slices with fewer users (Slice 1), while the more significant number of users perform worse than the average (Slice 2).

As identified in Figure 6.8, the slices' reconfiguration delay impacts the performance of some infrastructure processes, *e.g.*, service migrations. As the infrastructure does not

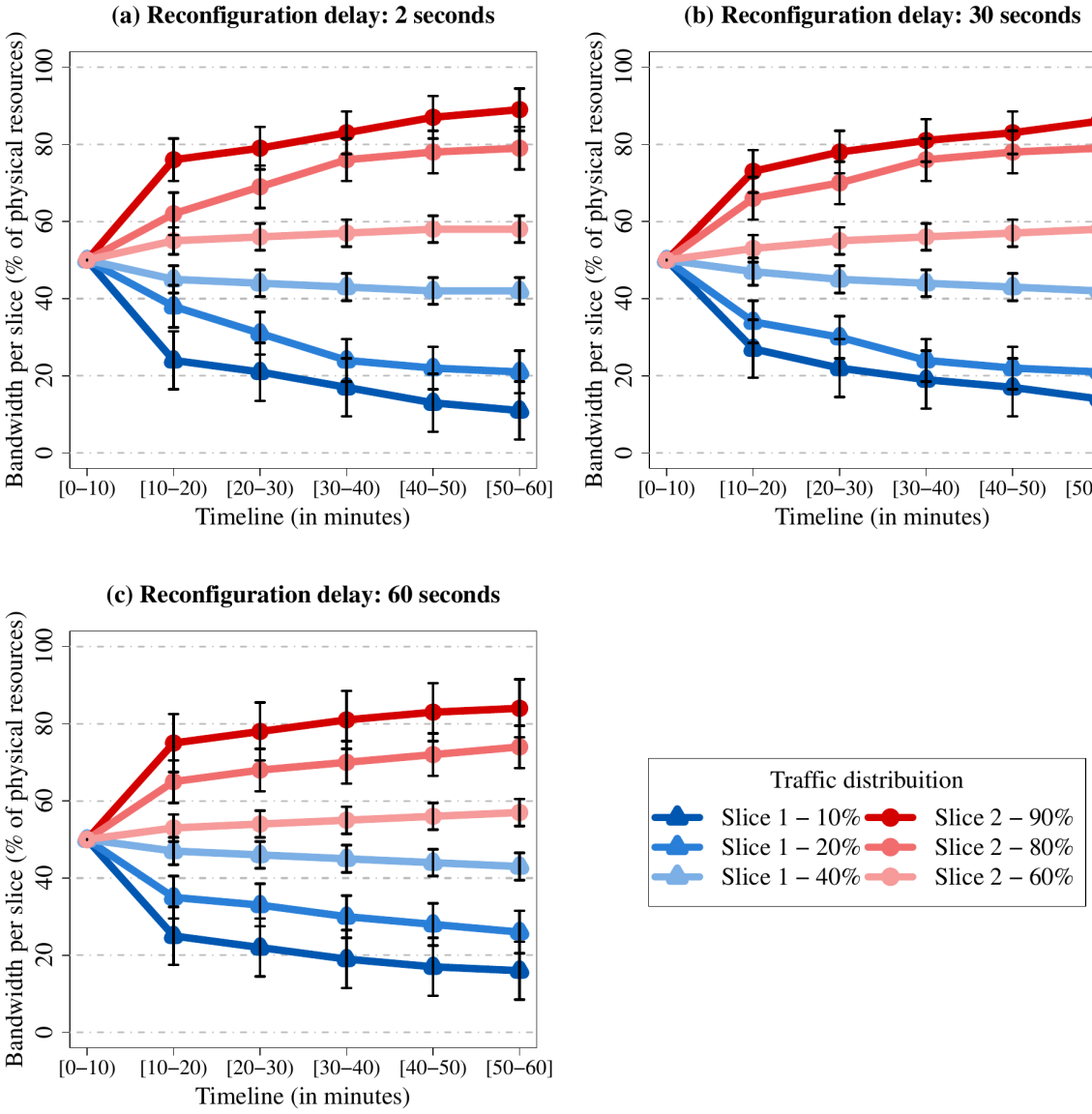


Figure 6.7: Timeline of bandwidth distribution among dynamic slices.

provide the proper number of migrations, it may leave the services in fog nodes that may no longer fulfil their service requirements. In high reconfiguration delay scenarios, if users present a high level of mobility, the infrastructure may present difficulty in providing seamless mobility management strategies that face demand variations.

Once the services are left in nodes that do not provide the best environment to their users anymore, the users may face some decrease in service quality. Figure 6.9 presents the impacts of different slice reconfiguration delays in service latency due to different service migration frequencies.

It can be identified that, while Slice 2 in scenarios with higher reconfiguration delay presents lower latency, the remainder slices (Slice 1), which serve the more significant number of users, present latency worse than the expected average performance. Slices that quickly balance their resources, *e.g.*, Figure 6.9 (a), provide a better and more stable service latency to its users (2.8 variance) instead of the users in the Figure 6.9 (c) (4.3

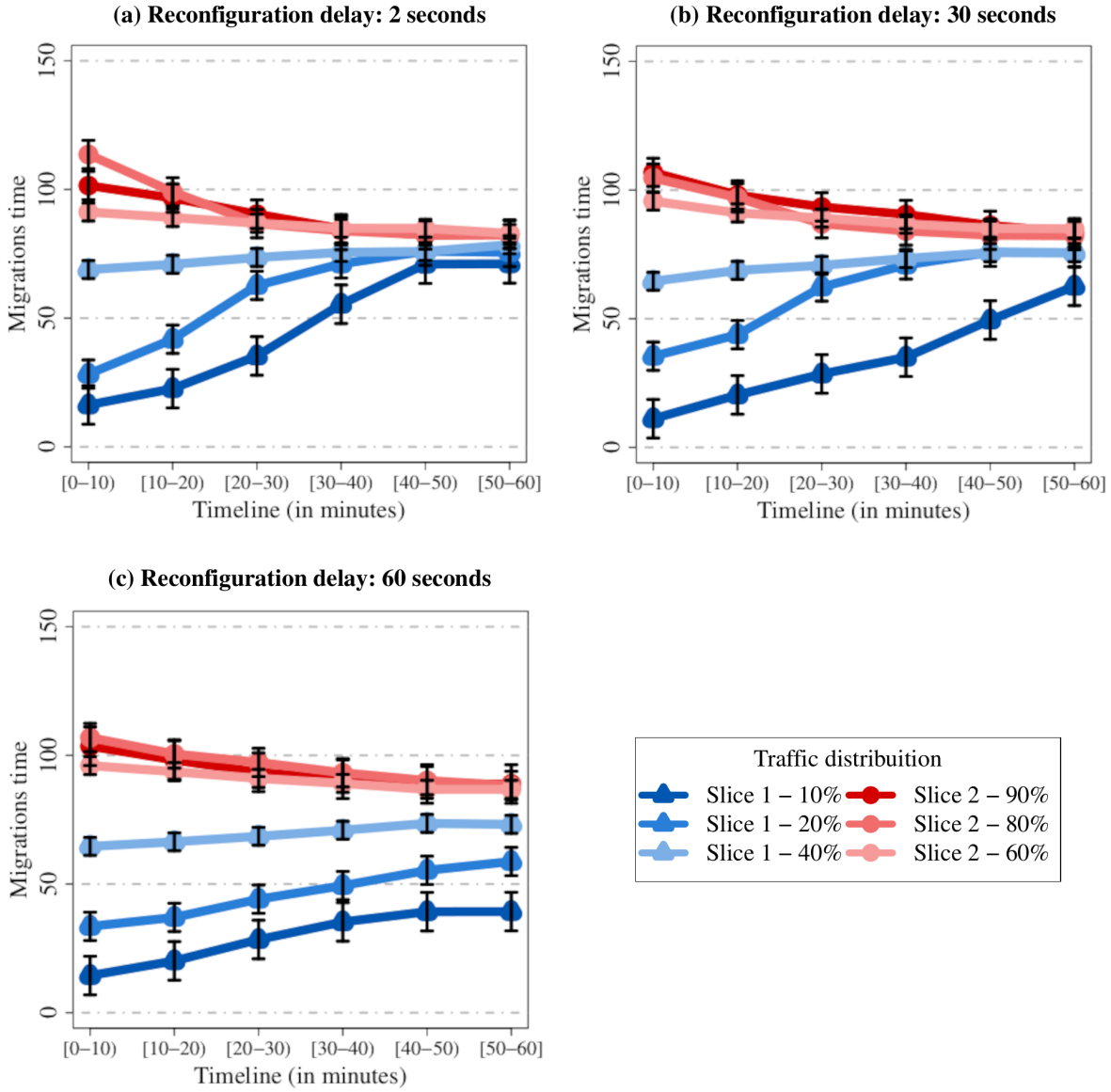


Figure 6.8: Timeline of the time required to perform service migrations.

variance).

This experiment evaluating the impact of different values on the slice reconfiguration delay indicates that lower values lead the slices to better and more stable resource distributions. Consequently, those slices provide a better environment to serve users' service requirements.

6.4 Chapter Conclusions

Dynamic network slicing allows the network infrastructures to change their characteristics in run time. However, most works do not evaluate how computing overhead impacts network performance. This chapter presented one mathematical model that estimates the required time to allocate one slice instance. Based on that delay, the impact of

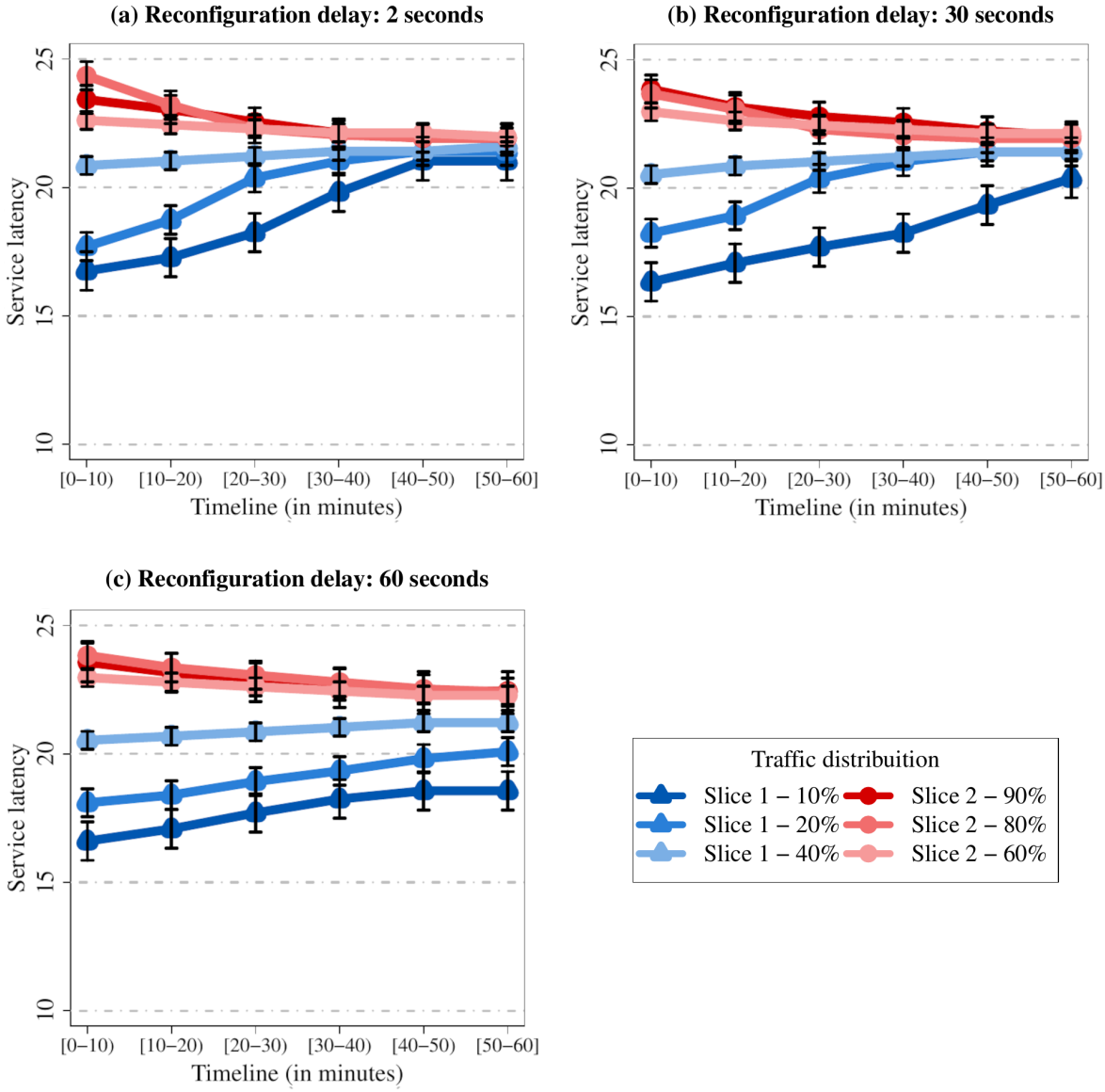


Figure 6.9: Timeline of service latency.

different allocation delays was simulated in realistic scenarios to evaluate their impact on the network performance.

Regarding the impact of reallocations in the slice load balance capabilities, the results showed that: 1) in a 60 minutes lifecycle scenario, slices that perform faster reconfigurations (2 seconds) have, on average, 78.1% of their reconfigurations in the first 30 minutes of their lifecycle; meanwhile, slices with slower reconfigurations (60 seconds) have 71.9%. In that scenario, faster reconfigurations tend to quicker converge their resources to a stable level that is suitable for their users' demand; 2) slices that quickly balance their resources (2 seconds reconfiguration delay) provide lower and more stable latency to their users (2.8 variance) than slices with 60 seconds delay (4.3 variance).

In Chapter 4, this thesis shows simulations of scenarios needed to evaluate the network slicing performance. In Chapter 5, an extensive evaluation of the performance of dynamic network slicing in the face of scenarios of mobile user support is presented. In addition,

this chapter presents one study that evaluates the trade-off of performing dynamic slicing. Based on those contributions, new resource management solutions can be developed based on the insights provided by this thesis. Some of those research directions are discussed in the following chapter.

Chapter 7

Conclusion

This chapter summarises the thesis's contributions and the research outcomes. Furthermore, this chapter discusses potential future research directions.

This thesis presented resource management solutions for dynamic network slicing orchestration in edge computing environments, in special, regarding user mobility support. In particular, those mechanisms focus on dynamic network slicing allocation, and service placement and migration. In summary, this work presented:

1. One heuristic solution which uses dynamic network slicing as a supporting technology to improve the service migration process in Chapter 4;

2. One study of the dynamic network slicing as a supporting technology for user mobility support in fog/edge computing infrastructures and vehicular networks in Chapter 5;

3. A mathematical model which describes the overhead/expected time to allocate and reconfigure the network slices in Chapter 6;

4. Resource management mechanisms for dynamic slicing allocation, which include solutions for defining the slice shaping, resource allocation and reallocation, and resource sharing scope in Chapter 5;

5. One simulator for slicing validation in fog computing-based infrastructures, including support to user mobility and service migration in Chapter 4.

The remainder of this chapter is organised as follows:

- Section 7.1 presents a summarised discussion regarding the thesis contribution, which includes answering seven research questions raised in this work and discussing the developed software derived from the content in this thesis;

- Section 7.2 presents two future research directions based on contributions presented in this thesis.

7.1 Thesis summary and contributions

5G networks aim to provide network resources for different user groups. In that context, the network must accommodate applications and services with diverse and sometimes conflicting requirements in parallel to serving users with different characteristics, such as mobility. Virtualisation technologies allow 5G's physical resources to be virtualised and presented as virtual networks. That process, named network slicing, creates a one-demand virtual network (slice) for each user group. Due to demand variations caused by user mobility, the 5G infrastructure needs to dynamically update its resource distribution among services and slices. Some challenges arise in that scenario, such as defining which resources should be allocated for each slice and defining when and where to perform a service migration to keep the service as close as possible to its users. Everything is made in parallel, considering mobility management requirements and other trade-off scenarios.

Based on that scenario, this work evaluated the performance of resource management mechanisms for two relevant challenges related to the network slicing context: network slicing allocation and service migration. This thesis raised six research questions, which were answered during the development of this work. The contribution of this work provides a foundation for further understanding the strengths and weaknesses points of network slicing technology for user mobility support. The research questions and the contribution of this work are described below.

- *What is the impact of resource sharing in the performance of network slicing for mobile users?* In Section 5.1.1, resource distribution and isolation in network slicing allocation were investigated. By using network slicing, it is possible to allocate more resources to slices that prioritise certain groups of users, *e.g.*, critical applications such as traffic control systems, military operations, and surgery procedures. In that experiment, the physical resources can be split into two virtual networks using network slicing. Different combinations of load and resource distribution were considered. The network slices were evaluated based on outage periods and service migration performance. In the experiment, high-priority slices required a shorter time to perform service migrations, presenting a reduced outage period. However, low-priority slices may receive unsatisfying resources in increasing demand periods in static resource allocation. Efficient resource distribution approaches need to be applied.

- *Can dynamic network slicing improve the performance of the service placement and migration process?* In Section 5.1, two different network slice allocation policies were evaluated in the context of service migration: static and dynamic allocation. Once the static network slicing approach fixes the amount of resources along the entire slice life-cycle, it might present under-utilisation and overcapacity situations. However, dynamic network slicing surpasses the inefficient allocation of resources in traffic variation scenarios. Once a service migration is requested, the NFV management and orchestration system can temporarily provide more resources to a slice deal with that traffic variation. Results show that dynamic slicing reduces under-utilisation scenarios by applying resource distribution on demand while improving

overall service migration performance. In the context of service migration, dynamic network slicing reduces the required time to perform service migrations and service outages, allows more service migrations if needed, and provides lower latency by placing services closer to their users.

- *What is the required time to perform network slice operations such as resource allocation and slice reconfigurations, and how does it impact the network performance?* Although Chapter 5 and several works present in the literature highlight the benefits of dynamic network slicing mechanisms, there was a scarcity of works researching weakness points and potential trade-off scenarios regarding the overhead of resource allocation operations in that context. In this work, Chapter 6 presents further research regarding the overhead present in dynamic network slicing operations. This work proposed one mathematical model that estimates the required time to perform the resource allocation operations of one slice instance, including onboarding, deployment and runtime operations. The proposed model includes VM and VNF allocation and boot-up among those operations. The impact of network performance was evaluated based on the required time to conclude such operations. In the simulated scenarios, which vary from 2 to 60 seconds to conclude one slice reconfiguration, it concludes that network slices that provide faster reconfigurations tend to converge their resources to a stable level suitable for their users' demand. As a result, slices that quickly balance their resources provide lower and more stable latency to their users than slices with higher reconfiguration delays.

- *What is the impact of user mobility in network slicing management?* In Section 5.2, different simulation scenarios were conducted to evaluate the performance of static and dynamic network slicing allocation using a realistic user mobility pattern in a larger-scale infrastructure. Based on the Luxembourg map and traffic patterns, those two slice resource allocation approaches were evaluated in an edge computing infrastructure. In addition, Chapter 6 presents the overhead of slice reconfigurations in that scenario. In general, user mobility triggers spatial and temporal traffic variations over time. In this context, such a dynamic scenario may present frequent and sometimes drastic traffic variations. On the one hand, dynamic resource allocation can redistribute resources along slices which present changing demands. However, as discussed in Chapter 6, if the time required to perform such slice modifications is high (30 seconds or longer), the infrastructure might not provide sufficient resource modification frequency to minimise resource underutilisation and maximise network performance. In conclusion, the frequency of drastic traffic variations triggered by user mobility might be too high to extract the full potential of dynamic resource distribution.

- *Is dynamic slicing a suitable solution to deal with traffic variation triggered by mobile users in edge computing environments?* In summary, based on results presented in Chapter 5 and Chapter 6, dynamic network slicing has the capability of i) setting priority levels for distinct user groups and providing them different resources, ii) reducing resource underutilisation and improves network performance by on-demand

resource (re)distributions, iii) improving service migrations by providing more resources to reduce the required time to perform that process. As a result, it allows the network to perform more migrations and keep its services closer to its users, reducing service latency. However, the cost of runtime slice modifications, in terms of processing demand and required time to perform that process, might affect the efficiency and frequency of the slice modification process.

- *Can dynamic slicing be applied in vehicular-cloud environments?* Extending the evaluation of dynamic network slicing in edge computing infrastructures, Section 5.3 includes vehicular clouds in that scenario. That work evaluates the network slice resource management performing vehicular clouds' resources as part of the end-to-end slice. Close vehicles, *i.e.*, within the same physical zone, compose a vehicular cloud and can share or request computing resources to/from nearby vehicles. In the experiment, the pool of resources that compose one slice could be composed of only edge resources, only vehicular, or a hybrid approach that combines vehicular and edge resources. Due to higher resource availability, results show that vehicular-only and hybrid scenarios presented the best results in terms of service latency. Results regarding the number of migrations and the required time to proceed with those operations do not show statistically significant improvements. It is worth mentioning that, due to their mobility, the vehicle's average permanency time in a zone is 4:26 minutes in the experiments. It increases the frequency in which the slice includes/excludes vehicles' resources from the set of available resources. Each reconfiguration incurs an overhead. On average, around 50 reconfigurations are performed within the 60-minute slice lifetime considered in the simulations. That slice reconfiguration frequency might present a high overhead in resource management and be impractical in larger and more dynamic scenarios.

- *Is there a validation environment to simulate network resource management scenarios, which includes user mobility, network slicing and service migration?* At the beginning of the development of this work, there was no state-of-the-art simulation environment for resource management research of edge computing environments, which included user mobility, service migration, and network slicing. As a contribution to this work, Chapter 4 introduces the MobFogSim simulator, which fulfils that gap. The development of this thesis presents as an outcome a free software simulator named MobFogSim[1]. MobFogSim supports user mobility, end-to-end dynamic network slicing and vehicular network simulation in an edge-cloud computing architecture. This open-source validation environment for smart city scenarios is available for community use. At this moment, that software was downloaded more than one hundred times, and the paper that introduces that simulator presents more than one hundred citations. As an impact of the availability of such a software, for the best of our knowledge, at least 26 works report MobFogSim as the validation environment for its researches. It includes: twenty one conference and journal papers [38, 158, 81, 83, 162, 23, 39, 86, 82, 171, 9, 172, 8, 128, 170, 173, 84, 98, 169, 129, 98], one undergraduate thesis [120], two master thesis [136, 77] and one PhD thesis [124].

[1]https://github.com/diogomg/MobFogSim. Last accessed: April 10th, 2024

In conclusion, the work developed along this thesis provides as a legacy two main contributions that potentially can improve computer science: 1) answering different research questions regarding resource management of network slicing in fog computing environments in the context of user mobility support. By researching this topic, this work provides strengths and weaknesses points of network slicing technology regarding user mobility support, and 2) by producing an open source software as the validation environment for the evaluated scenario, this work provides a valuable tool for researching community bypass limitations regarding building real testbeds infrastructures. Different research directions can be based on contributions developed in this thesis. Some of them are further discussed in the following section.

7.2 Future Work

Machine learning research has been presented as a long-term trend within applicability in many fields, from finance to medicine [143]. Not different from those fields, machine learning is "a foundational enabler of 6G" and beyond [78], presenting impact on security [133], routing and transmission control [153], energy efficiency [58], resource management [13, 145, 126], and user mobility support [76, 175], to cite a few. In this section, machine learning is discussed from two points of view: as a means (ML for network resource management) and as an end (ML applications taking advantage of the network infrastructure) in the context of future research on user mobility support.

Based on contributions presented in this thesis, two future research directions are further discussed in this section: 1) On one hand, zero-touch networks can take advantage of machine learning as a means to develop the next-generation network automation; 2) On the other hand, aiming to support machine learning applications, parallelisation and hardware customisation is a way to improve the performance of those applications. Those research directions are further discussed below.

- *Zero-touch Networks and Computing Continuum:* The benefits of computing powerful centralised cloud data centres in combination with distributed low latency edge servers offer the best characteristics of each resource domain. However, integrating those multiple domains, from IoT devices through the edge and cloud, known as the computing continuum, is still challenging [110, 21]. In particular, future research regarding the computing continuum will be guided by the "dynamic allocation of the computation, and in particular of the execution of the Artificial Intelligence (AI), and the related resource orchestration, network partitioning and support for context awareness" [110]. Regarding user mobility support in network slicing, AI and ML "have been identified as key-technologies in this context, providing the means to efficiently automate the management of mobile network resources in response to real-time trigger events, *e.g.*, user mobility or traffic load variation" [174].

  In this context, machine learning-based management and orchestration systems can improve the efficiency and automation of network operations in the slice life cycle [126, 44, 157]. Zero-touch networks [46, 60], self-organising networks [57], self-distributed systems [131], and other machine learning-based solutions will have a

key role in the automation of large-scale networks and multi-domain resource management [72, 45]. ML solutions applied to network orchestration can unlock network management with minimal human interference [134].

In particular, regarding network slicing research development, "promoting self-optimisation and self-(re)configuration in E2E network slicing is required through intelligent resource orchestration" [45]. In this context, ML-based solutions can fully self-manage operations such as the prediction of resource demand and user requirements, the placement and migration of VNFs, and the creation, configuration, and management of network slices [134, 126]. However, the development of AI/ML techniques for slice resource management is still an open challenge [126]. Based on the contributions of this thesis, future research on machine learning solutions can be developed to automate network slice allocation and service placement.

- *Dynamic hardware customisation for mobile users:* Historically, computing power has been improved by increasing the number of transistors and threads [89]. However, in recent years, domain specialisation has received great attention. Hardware customised for a particular application avoids unnecessary resources and focuses on specific characteristics that improve the application's performance. Customised hardware trades off flexibility for efficiency, enabling applications to run faster, reducing energy consumption, or both [89] [22]. Hardware customisation could be key to improving application performance in edge devices.

  FPGA (field programmable gate array) [24, 22] technology is a potential key enabler in that scenario. A FPGA is composed of building blocks, *e.g.*, processor, memory and IO, connected by logic cells. The key advantage of FPGA is that the hardware configuration can be made dynamically and on demand by selecting which building blocks will be used. Attaching FPGAs into edge nodes could offer tailored hardware for machine learning applications at the edge [140, 32]. That hardware customisation in edge can improve the efficiency of ML implementations [74], accelerating "decision-making, hypothesis testing and even enable just-in-time interventions" [50]. In this context, the open-source HLS4ML framework [50] translates trained ML models into FPGA implementations using high-level synthesis (HLS) tools.

  Despite the lower computing power at the edge being mitigated by hardware customisation provided by FPGAs, providing that customisation in every edge node a mobile user connects is challenging. As future work, using the HLS4ML framework [50] to translate ML models into HLS projects, which describes customised settings for ML implementation in FPGA devices, such hardware configuration could be carried by users and applied in different edge nodes. Similarly to service/VM migrations in a follow-me cloud scenario, in this research line, the hardware settings could *follow* the mobile user. By carrying hardware settings, mobile users can take advantage of customised hardware anywhere to execute applications in edge in a performance-enhanced environment.

Bibliography

[1] 3GPP. System architecture for the 5G system (5GS). *3GPP TS 23.501 version 16.6.0 Release 16*, 2020.

[2] 3GPP. 5G end-to-end key performance indicators (KPI). *3GPP TS 28.554 version 19.0.0 Release 19*, 2024.

[3] 3GPP. 5G performance measurements. *3GPP TS 28.552 version 19.0.0 Release 19*, 2024.

[4] A. Dilmac, M. E. Gure, T. Tugcu. SliceSim: A simulation suite for network slicing in 5G networks, May 2019.

[5] Alaa Awad Abdellatif, Amr Mohamed, Aiman Erbad, and Mohsen Guizani. Dynamic network slicing and resource allocation for 5G-and-beyond networks. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 262–267. IEEE, 2022.

[6] Mohammed Salah Abood, Hua Wang, Dongxuan He, Ziqi Kang, and Agnes Kawoya. Intelligent network slicing in V2X networks–a comprehensive review. *Journal of Artificial Intelligence and Technology*, 3(2):75–84, 2023.

[7] Rami Akrem Addad, Diego Leonel Cadette Dutra, Tarik Taleb, and Hannu Flinck. AI-based network-aware service function chain migration in 5G and beyond networks. *IEEE Transactions on Network and Service Management*, 19(1):472–484, 2022.

[8] M. A. B. Al-Tarawneh. Mobility-aware container migration in cloudlet-enabled IoT systems using integrated muticriteria decision making. *International Journal of Advanced Computer Science and Applications*, 11(9), 2020.

[9] Mehbub Alam, Nurzaman Ahmed, Rakesh Matam, and Ferdous Ahmed Barbhuiya. L3Fog: fog node selection and task offloading framework for mobile IoT. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2022.

[10] Tanweer Alam. Cloud-based IoT applications and their roles in smart cities. *Smart Cities*, 4(3):1196–1219, 2021.

[11] NGMN Alliance. 5G end-to-end architecture framework. *Tech. Rep.*, pages 04–0ct, 2017.

[12] Thales T Almeida, Lucas de C Gomes, Fernando M Ortiz, JoséG R Júnior, and Luís HMK Costa. IEEE 802.11 p performance evaluation: simulations vs. real experiments. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3840–3845. IEEE, 2018.

[13] Sana Anjum, Deepak Upadhyay, Kamna Singh, and Prashant Upadhyay. Machine learning-based resource allocation algorithms for 6G networks. In *2024 2nd International Conference on Disruptive Technologies (ICDT)*, pages 1086–1091. IEEE, 2024.

[14] Ahmed Arbaoui, Sahraoui Abdelatif, and Makhlouf Derdour. Network slicing solutions for internet of vehicles (IoV) networks: A review. In *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–7. IEEE, 2024.

[15] Alia Asheralieva, Dusit Niyato, and Yoshikazu Miyanaga. Efficient dynamic distributed resource slicing in 6G multi-access edge computing networks with online admm and message passing graph neural networks. *IEEE Transactions on Mobile Computing*, 2023.

[16] GSM Association et al. Network slicing use case requirements, 2018.

[17] Miloud Bagaa, Diego Leonel Cadette Dutra, Tarik Taleb, and Hannu Flinck. Toward enabling network slice mobility to support 6G system. *IEEE Transactions on Wireless Communications*, 21(12):10130–10144, 2022.

[18] Salvador V Balkus, Honggang Wang, Brian D Cornet, Chinmay Mahabal, Hieu Ngo, and Hua Fang. A survey of collaborative machine learning using 5G vehicular communications. *IEEE Communications Surveys & Tutorials*, 24(2):1280–1303, 2022.

[19] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines. 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167:106984, 2020.

[20] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. SUMO – simulation of urban mobility: An overview. In *3rd International Conference on Advances in System Simulation (SIMUL)*, pages 55–60. ThinkMind, October 2011.

[21] Luiz Bittencourt, Roger Immich, Rizos Sakellariou, Nelson Fonseca, Edmundo Madeira, Marilia Curado, Leandro Villas, Luiz DaSilva, Craig Lee, and Omer Rana. The internet of things, fog and cloud continuum: Integration and challenges. *Internet of Things*, 3:134–155, 2018.

[22] Christophe Bobda, Joel Mandebi Mbongue, Paul Chow, Mohammad Ewais, Naif Tarafdar, Juan Camilo Vega, Ken Eguro, Dirk Koch, Suranga Handagala, Miriam Leeser, et al. The future of FPGA acceleration in datacenters and the cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 15(3):1–42, 2022.

[23] Ruslan Borysov. Development of functionalities extension approach and implementation of address routing for IFogSim based simulators. *Technology audit and production reserves*, 3(2):59, 2021.

[24] Andrew Boutros and Vaughn Betz. FPGA architecture: Principles and progression. *IEEE Circuits and Systems Magazine*, 21(2):4–29, 2021.

[25] Haotong Cao, Jianbo Du, Haitao Zhao, Daniel Xiapu Luo, Neeraj Kumar, Longxiang Yang, and F Richard Yu. Toward tailored resource allocation of slices in 6G networks with softwarization and virtualization. *IEEE Internet of Things Journal*, 9(9):6623–6637, 2021.

[26] Mohammed Chahbar, Gladys Diaz, Abdulhalim Dandoush, Christophe Cérin, and Kamal Ghoumid. A comprehensive survey on the E2E 5G network slicing model. *IEEE Transactions on Network and Service Management*, 18(1):49–62, 2020.

[27] Chen Chen, Bin Liu, Shaohua Wan, Peng Qiao, and Qingqi Pei. An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1840–1852, 2020.

[28] Asma Chiha, Marlies Van der Wee, Didier Colle, and Sofie Verbrugge. Network slicing cost allocation model. *Journal of Network and Systems Management*, 28:627–659, 2020.

[29] Sihyun Choi, Siyoung Choi, Goodsol Lee, Sung-Guk Yoon, and Saewoong Bahk. Deep reinforcement learning for scalable dynamic bandwidth allocation in RAN slicing with highly mobile users. *IEEE Transactions on Vehicular Technology*, 2024.

[30] Stuart Clayman, Augusto Neto, Fábio Verdi, Sand Correa, Silvio Sampaio, Ilias Sakelariou, Lefteris Mamatas, Rafael Pasquini, Kleber Cardoso, Francesco Tusa, et al. The NECOS approach to end-to-end cloud-network slicing as a service. *IEEE Communications Magazine*, 59(3):91–97, 2021.

[31] L. Codeca, R. Frank, and T. Engel. Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. In *IEEE Conference on Vehicular Networking (VNC)*, pages 1–8, Dec. 2015.

[32] Jason Cong, Jason Lau, Gai Liu, Stephen Neuendorffer, Peichen Pan, Kees Vissers, and Zhiru Zhang. FPGA HLS today: successes, challenges, and opportunities. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 15(4):1–42, 2022.

[33] J. Cosmas, N. Jawad, M. Salih, S. Redana, and O. Bulakci. 5G PPP Architecture Working Group view on 5G architecture, 2019.

[34] Antonio ATR Coutinho, Elisangela O Carneiro, and Fabíola Greve. Simulation and modeling tools for fog computing. In *Fog Computing*, pages 51–84. Chapman and Hall/CRC, 2022.

[35] Yaping Cui, Hongji Shi, Ruyan Wang, Peng He, Dapeng Wu, and Xinyun Huang. Multi-agent reinforcement learning for slicing resource allocation in vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[36] Shuping Dang, Osama Amin, Basem Shihada, and Mohamed-Slim Alouini. What should 6G be? *Nature Electronics*, 3(1):20–29, 2020.

[37] Fadoua Debbabi, Rihab Jmal, Lamia Chaari Fourati, and Rui Luis Aguiar. An overview of interslice and intraslice resource allocation in B5G telecommunication networks. *IEEE Transactions on Network and Service Management*, 19(4):5120–5132, 2022.

[38] D Deepa and K. R. Jothi. Enhancing secure mutual authentication in mobile fog computing environment. In *International Conference on Advanced computing Technologies & Applications (ICACTA)*, pages 1–4. IEEE, 2022.

[39] D Deepa and KR Jothi. Classification of request-based mobility load balancing in fog computing. *Comput. Syst. Sci. Eng.*, 46(1):137–151, 2023.

[40] Xiaoheng Deng, Leilei Wang, Jinsong Gui, Ping Jiang, Xuechen Chen, Feng Zeng, and Shaohua Wan. A review of 6G autonomous intelligent transportation systems: Mechanisms, applications and challenges. *Journal of Systems Architecture*, page 102929, 2023.

[41] Hamza Djigal, Jia Xu, Linfeng Liu, and Yan Zhang. Machine and deep learning for resource allocation in multi-access edge computing: A survey. *IEEE Communications Surveys & Tutorials*, 24(4):2449–2494, 2022.

[42] Pedro F do Prado, Maycon LM Peixoto, Marcelo C Araújo, Eduardo S Gama, Diogo M Gonçalves, Matteus VS Silva, Roger Immich, Edmundo RM Madeira, and Luiz F Bittencourt. Mobile edge computing for content distribution and mobility support in smart cities. *Mobile Edge Computing*, pages 473–500, 2021.

[43] Afra Domeke, Bruno Cimoli, and Idelfonso Tafur Monroy. Integration of network slicing and machine learning into edge networks for low-latency services in 5G and beyond systems. *Applied Sciences*, 12(13):6617, 2022.

[44] Adnei Donatti, Sand L Correa, Joberto SB Martins, Antonio JG Abelem, Cristiano Bonato Both, Flávio de Oliveira Silva, José A Suruagy, Rafael Pasquini, Rodrigo Moreira, Kleber V Cardoso, et al. Survey on machine learning-enabled network slicing: Covering the entire life cycle. *IEEE Transactions on Network and Service Management*, 21(1):994–1011, 2023.

[45] Sina Ebrahimi, Faouzi Bouali, and Olivier CL Haas. Resource management from single-domain 5G to end-to-end 6G network slicing: A survey. *IEEE Communications Surveys & Tutorials*, 2024.

[46] Mirna El Rajab, Li Yang, and Abdallah Shami. Zero-touch networks: Towards next-generation network automation. *Computer Networks*, page 110294, 2024.

[47] Ahmet M Elbir, Burak Soner, Sinem Çöleri, Deniz Gündüz, and Mehdi Bennis. Federated learning in vehicular networks. In *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pages 72–77. IEEE, 2022.

[48] Ali Esmaeily and Katina Kralevska. Small-scale 5G testbeds for network slicing deployment: A systematic review. *Wireless Communications and Mobile Computing*, 2021(1):6655216, 2021.

[49] Jose Jurandir Alves Esteves, Amina Boubendir, Fabrice Guillemin, and Pierre Sens. A heuristically assisted deep reinforcement learning approach for network slice placement. *IEEE Transactions on Network and Service Management*, 19(4):4794–4806, 2022.

[50] Farah Fahim, Benjamin Hawks, Christian Herwig, James Hirschauer, Sergo Jindariani, Nhan Tran, Luca P Carloni, Giuseppe Di Guglielmo, Philip Harris, Jeffrey Krupa, et al. Hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices. *arXiv preprint arXiv:2103.05579*, 2021.

[51] Muhammad Fahimullah, Guillaume Philippe, Shohreh Ahvar, and Maria Trocan. Simulation tools for fog computing: a comparative analysis. *Sensors*, 23(7):3492, 2023.

[52] Jinliang Fan and Mostafa H Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12. Citeseer, 2006.

[53] Haoxian Feng, Zhaogang Shu, Tarik Taleb, Yuantao Wang, and Zhiwei Liu. An aggressive migration strategy for service function chaining in the core cloud. *IEEE Transactions on Network and Service Management*, 20(2):2025–2039, 2022.

[54] J. Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu. Dynamic network slicing and resource allocation in mobile edge computing systems. *IEEE Transactions on Vehicular Technology*, 69(7):7863–7878, 2020.

[55] Aidin Ferdowsi, Ursula Challita, and Walid Saad. Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview. *ieee vehicular technology magazine*, 14(1):62–70, 2019.

[56] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, 2013.

[57] Hasna Fourati, Rihab Maaloul, Lamia Chaari, and Mohamed Jmaiel. Comprehensive survey on self-organizing cellular network approaches applied to 5G networks. *Computer Networks*, 199:108435, 2021.

[58] Tulsi Pawan Fowdur and Bhuvaneshwar Doorgakant. A review of machine learning techniques for enhanced energy efficient 5G and 6G communications. *Engineering Applications of Artificial Intelligence*, 122:106032, 2023.

[59] Leandro A Freitas, Vinicius G Braga, Sand L Corrêa, Lefteris Mamatas, Christian E Rothenberg, Stuart Clayman, and Kleber V Cardoso. Slicing and allocation of transformable resources for the deployment of multiple virtualized infrastructure managers (VIMS). In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 424–432. IEEE, 2018.

[60] Jorge Gallego-Madrid, Ramon Sanchez-Iborra, Pedro M Ruiz, and Antonio F Skarmeta. Machine learning-based zero-touch network and service management: A survey. *Digital Communications and Networks*, 8(2):105–123, 2022.

[61] Nihal Gaouar and Mohamed Lehsaini. Toward vehicular cloud/fog communication: A survey on data dissemination in vehicular ad hoc networks using vehicular cloud/fog computing. *International Journal of Communication Systems*, 34(13):e4906, 2021.

[62] Gines Garcia-Aviles, Marco Gramaglia, Pablo Serrano, Francesco Gringoli, Sergio Fuente-Pascual, and Ignacio Labrador Pavon. Experimenting with open source tools to deploy a multi-service and multi-slice mobile network. *Computer Communications*, 150:1–12, 2020.

[63] Amir Gharehgoli, Ali Nouruzi, Nader Mokari, Paeiz Azmi, Mohammad Reza Javan, and Eduard A Jorswieck. AI-based resource allocation in end-to-end network slicing under demand and CSI uncertainties. *IEEE Transactions on Network and Service Management*, 2023.

[64] Monika Gill and Dinesh Singh. A comprehensive study of simulation frameworks and research directions in fog computing. *Computer Science Review*, 40:100391, 2021.

[65] D. Gonçalves, L. Bittencourt, and E. Madeira. Análise da predição de mobilidade na migração de aplicações em computação em névoa. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 580–593. SBC, 2019.

[66] D. Gonçalves, C. Puliafito, E. Mingozzi, O. Rana, L. Bittencourt, and E. Madeira. Dynamic network slicing in fog computing for mobile users in MobFogSim. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 237–246. IEEE, 2020.

[67] Diogo M Gonçalves, Luiz F Bittencourt, and Edmundo RM Madeira. Fatiamento dinâmico de redes em computaçao em névoa para usuários móveis. In *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 57–70. SBC, 2021.

[68] Diogo M Gonçalves, Luiz F Bittencourt, and Edmundo RM Madeira. Alocaçao de fatias de rede fim-a-fim para usuários móveis utilizando o simulador MobFogSim. In *Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 112–125. SBC, 2022.

[69] Diogo M Gonçalves, Luiz F Bittencourt, and Edmundo RM Madeira. Overhead and performance of dynamic network slice allocation for mobile users. *Future Generation Computer Systems*, 2024.

[70] Diogo M Gonçalves, Carlo Puliafito, Enzo Mingozzi, Luiz F Bittencourt, and Edmundo RM Madeira. End-to-end network slicing in vehicular clouds using the MobFogSim simulator. *Ad Hoc Networks*, page 103096, 2023.

[71] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, May 2017.

[72] Mohammad Asif Habibi, Bin Han, Amina Fellan, Wei Jiang, Adrián Gallego Sánchez, Ignacio Labrador Pavón, Amina Boubendir, and Hans D Schotten. Towards an open, intelligent, and end-to-end architectural framework for network slicing in 6G communication systems. *IEEE Open Journal of the Communications Society*, 2023.

[73] Wafa Hamdi, Chahrazed Ksouri, Hasan Bulut, and Mohamed Mosbah. Network slicing based learning techniques for IoV in 5G and beyond networks. *IEEE Communications Surveys & Tutorials*, 2024.

[74] Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. Edge computing with artificial intelligence: A machine learning perspective. *ACM Computing Surveys*, 55(9):1–35, 2023.

[75] Johanna Andrea Hurtado Sánchez, Katherine Casilimas, and Oscar Mauricio Caicedo Rendon. Deep reinforcement learning for resource management on network slicing: A survey. *Sensors*, 22(8):3031, 2022.

[76] Hatem Ibn-Khedher, Mohammed Laroui, Mohammed Alfaqawi, Ahlam Magnouche, Hassine Moungla, and Hossam Afifi. 6G-edge support of internet of autonomous vehicles: A survey. *Transactions on Emerging Telecommunications Technologies*, 35(1):e4918, 2024.

[77] Abdul'Rauf Ijaoba. Temporal logic algorithms for multiple users and services in mobile edge computing. Master's thesis, Dublin, National College of Ireland, 2021.

[78] Aqeel Thamer Jawad, Rihab Maaloul, and Lamia Chaari. A comprehensive survey on 6G and beyond: Enabling technologies, opportunities of machine learning and challenges. *Computer Networks*, page 110085, 2023.

[79] Devki Nandan Jha, Khaled Alwasel, Areeb Alshoshan, Xianghua Huang, Ranesh Kumar Naha, Sudheer Kumar Battula, Saurabh Garg, Deepak Puthal, Philip James, Albert Zomaya, Schahram Dustdar, and Rajiv Ranjan. IoTSim-Edge: A simulation framework for modeling the behavior of internet of things and edge computing environments. *Software: Practice and Experience*, 50(6):844–867, Jan. 2020.

[80] Luyue Ji, Shibo He, Wenjie Wu, Chaojie Gu, Jichao Bi, and Zhiguo Shi. Dynamic network slicing orchestration for remote adaptation and configuration in industrial IoT. *IEEE Transactions on Industrial Informatics*, 18(6):4297–4307, 2022.

[81] Abin Gigo Joseph, Mihir Madhav Gokhale, Joseph Mani Jacob Mani, and T Veni. Mobility aware computation offloading in fog devices using virtual machine migration. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2023.

[82] Sangeeta Kakati, Mehbub Alam, Rakesh Matam, Ferdous Ahmed Barbhuiya, and Mithun Mukherjee. Mobility-aware task offloading in fog-assisted networks. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 2897–2902. IEEE, 2022.

[83] Navjeet Kaur, Ashok Kumar, and Rajesh Kumar. PROMO: PROactive MObility-support model for task scheduling in fog computing. *International Journal of Computers and Applications*, 44(11):1092–1101, 2022.

[84] Navjeet Kaur, Ayush Mittal, Umesh Kumar Lilhore, Sarita Simaiya, Surjeet Dalal, and Yogesh Kumar Sharma. An adaptive mobility-aware secure handover and scheduling protocol for earth observation (EO) communication using fog computing. *Earth Science Informatics*, pages 1–18, 2024.

[85] G. Kecskemeti. DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, 58:188–218, 2015.

[86] Bijendra Kumar and Vibha Jain. A comparative study on simulation environment on fog computing. In *2021 Emerging Trends in Industry 4.0 (ETI 4.0)*, pages 1–8. IEEE, 2021.

[87] Viswanath KumarSkandPriya, Abdulhalim Dandoush, and Gladys Diaz. Slicenet: a simple and scalable flow-level simulator for network slice provisioning and management. In *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE, 2023.

[88] Chun Sing Lai, Youwei Jia, Zhekang Dong, Dongxiao Wang, Yingshan Tao, Qi Hong Lai, Richard TK Wong, Ahmed F Zobaa, Ruiheng Wu, and Loi Lei Lai. A review of technical standards for smart cities. *Clean Technologies*, 2(3):290–310, 2020.

[89] Charles E Leiserson, Neil C Thompson, Joel S Emer, Bradley C Kuszmaul, Butler W Lampson, Daniel Sanchez, and Tao B Schardl. There's plenty of room

at the top: What will drive computer performance after Moore's law? *Science*, 368(6495):eaam9744, 2020.

[90] I. Lera, C. Guerrero, and C. Juiz. YAFS: A simulator for IoT scenarios in fog computing. *IEEE Access*, 7:91745–91758, 2019.

[91] J. Li, L. Chen, and J. Chen. Enabling technologies for low-latency service migration in 5G transport networks. *Journal of Optical Communications and Networking*, 13(2):A200–A210, 2021.

[92] M. M. Lopes, W. A. Higashino, M. A.M. Capretz, and L. F. Bittencourt. MyiFogSim: A simulator for virtual machine migration in fog computing. In *ACM 6th International Workshop on Clouds and (eScience) Applications Management (CloudAM). Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, pages 47–52, 2017.

[93] Qiu Lu, Gaoxing Li, and Hengzhou Ye. EdgeSim++: A realistic, versatile, and easily customizable edge computing simulator. *IEEE Internet of Things Journal*, 2024.

[94] Md Julkar Nayeen Mahi, Sudipto Chaki, Shamim Ahmed, Milon Biswas, M Shamim Kaiser, Mohammad Shahidul Islam, Mehdi Sookhak, Alistair Barros, and Md Whaiduzzaman. A review on VANET research: Perspective of recent emerging technologies. *IEEE Access*, 10:65760–65783, 2022.

[95] Redowan Mahmud, Samodha Pallewatta, Mohammad Goudarzi, and Rajkumar Buyya. IFogSim2: An extended IFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Elsevier Journal of Systems and Software*, 190, Aug. 2022.

[96] K. Makhijani, J. Qin, R. Ravindran, L. Geng, L. Qiang, S. Peng, X. de Foy, A. Rahman, and A. Galis. Network slicing use cases: Network customization and differentiated services. *draft-netslices-usecases-02 (Work in Progress)*, 2017.

[97] Dimitrios Michael Manias and Abdallah Shami. The need for advanced intelligence in NFV management and orchestration. *IEEE Network*, 35(1):365–371, 2020.

[98] Tahir Maqsood, Faisal Rehman, Saad Mustafa, Muhammad Amir Khan, Neelam Gohar, Abeer D Algarni, Hela Elmannai, et al. Content caching in mobile edge computing based on user location and preferences using cosine similarity and collaborative filtering. *Electronics (2079-9292)*, 12(2), 2023.

[99] R Deiny Mardian, Muhammad Suryanegara, and Kalamullah Ramli. Measuring quality of service (QoS) and quality of experience (QoE) on 5G technology: A review. In *2019 IEEE International Conference on Innovative Research and Development (ICIRD)*, pages 1–6. IEEE, 2019.

[100] S. V. Margariti, V. V. Dimakopoulos, and G. Tsoumanis. Modeling and simulation tools for fog computing — a comprehensive survey from a cost perspective. *Future Internet*, 12(5):89, 2020.

[101] A. Markus and A. Kertesz. A survey and taxonomy of simulation environments modelling fog computing. *Simulation Modelling Practice and Theory*, 101:102042, 2020.

[102] A. Markus and A. Kertesz. Investigating IoT application behaviour in simulated fog environments. *Cloud Computing and Services Science*, 1399:258, 2021.

[103] András Márkus. *DISSECT-CF-Fog: A Simulation Environment for Analysing the Cloud-to-Thing Continuum.* PhD thesis, Szegedi Tudomanyegyetem (Hungary), 2023.

[104] Jorge Martín-Pérez, Francesco Malandrino, Carla Fabiana Chiasserini, Milan Groshev, and Carlos J Bernardos. KPI guarantees in network slicing. *IEEE/ACM Transactions on Networking*, 2021.

[105] Charafeddine Mechalikh, Hajer Taktak, and Faouzi Moussa. PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments. *Computer Science and Information Systems*, 18(1):43–66, 2021.

[106] Jie Mei, Xianbin Wang, and Kan Zheng. Semi-decentralized network slicing for reliable V2V service provisioning: A model-free deep reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):12108–12120, 2022.

[107] Prodromos-Vasileios Mekikis, Kostas Ramantas, Angelos Antonopoulos, Elli Kartsakli, Luis Sanabria-Russo, Jordi Serra, David Pubill, and Christos Verikoukis. NFV-enabled experimental platform for 5G tactile internet support in industrial environments. *IEEE Transactions on Industrial Informatics*, 16(3):1895–1903, 2020.

[108] Oliver Michel, Roberto Bifulco, Gábor Rétvári, and Stefan Schmid. The programmable data plane: Abstractions, architectures, algorithms, and applications. *ACM Computing Surveys (CSUR)*, 54(4):1–36, 2021.

[109] Zhao Ming, Hao Yu, and Tarik Taleb. Federated deep reinforcement learning for prediction-based network slice mobility in 6G mobile networks. *IEEE Transactions on Mobile Computing*, 2024.

[110] Sergio Moreschini, Fabiano Pecorelli, Xiaozhou Li, Sonia Naz, David Hästbacka, and Davide Taibi. Cloud continuum: The definition. *IEEE Access*, 10:131876–131886, 2022.

[111] Anuja Nair and Sudeep Tanwar. Resource allocation in V2X communication: State-of-the-art and research challenges. *Physical Communication*, page 102351, 2024.

[112] Aadharsh Roshan Nandhakumar, Ayush Baranwal, Priyanshukumar Choudhary, Muhammed Golec, and Sukhpal Singh Gill. Edgeaisim: A toolkit for simulation and modelling of ai models in edge computing environments. *Measurement: Sensors*, 31:100939, 2024.

[113] Giovanni Nardini, Dario Sabella, Giovanni Stea, Purvi Thakkar, and Antonio Virdis. Simu5G – an OMNeT++ library for end-to-end performance evaluation of 5G networks. *IEEE Access*, 8:181176–181191, Oct. 2020.

[114] Almuthanna Nassar and Yasin Yilmaz. Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities. *IEEE Internet of Things Journal*, 9(1):222–235, 2022.

[115] Z. Ning, J. Huang, and X. Wang. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications*, 26(1):87–93, 2019.

[116] Sharnil Pandya, Gautam Srivastava, Rutvij Jhaveri, M Rajasekhara Babu, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, Spyridon Mastorakis, Md Jalil Piran, and Thippa Reddy Gadekallu. Federated learning for smart cities: A comprehensive survey. *Sustainable Energy Technologies and Assessments*, 55:102987, 2023.

[117] Gabriela Pereyra, Claudina Rattaro, and Pablo Belzarena. A 5G multi-slice cell capacity framework. In *ACM SIGCOMM 2021 Networking Networking Women Professional Development Workshop (N2Women'21)*, 2021.

[118] Gabriela Pereyra, Claudina Rattaro, and Pablo Belzarena. Py5cheSim: a 5G multi-slice cell capacity simulator. In *2021 XLVII Latin American Computing Conference (CLEI)*, pages 1–8. IEEE, 2021.

[119] Nattakorn Promwongsa, Amin Ebrahimzadeh, Roch H Glitho, and Noel Crespi. Joint VNF placement and scheduling for latency-sensitive services. *IEEE Transactions on Network Science and Engineering*, 9(4):2432–2449, 2022.

[120] Pietro Ruy Pugliesi and Edmundo Madeira. Simulação de modelos de provisionamento e migração de recursos nas bordas da internet utilizando o simulador MobFogSim. 2022.

[121] C. Puliafito, D. M. Goncalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. Bittencourt. MobFogSim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory*, 101:102062, 2020.

[122] C. Puliafito, E. Mingozzi, and G. Anastasi. Fog computing for the internet of mobile things: issues and challenges. In *Smart Computing (SMARTCOMP), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[123] C. Puliafito, C. Vallati, E. Mingozzi, G. Merlino, F. Longo, and A. Puliafito. Container migration in the fog: A performance evaluation. *MDPI Sensors*, 19(7), Mar. 2019.

[124] Carlo Puliafito et al. Companion fog computing: Migrating the fog service to support IoT device mobility. 2020.

[125] T. Qayyum, A. W. Malik, M. A. k. Khattak, O. Khalid, and S. U. Khan. FogNetSim++: A toolkit for modelling and simulation of distributed fog environment. *IEEE Access*, 6:63570–63583, Oct. 2018.

[126] Wajid Rafique, Joyeeta Barai, Abraham O Fapojuwo, and Diwakar Krishnamurthy. A survey on beyond 5G network slicing for smart cities applications. *IEEE Communications Surveys & Tutorials*, 2024.

[127] Preeti Rani and Rohit Sharma. Intelligent transportation system for internet of vehicles based vehicular networks for smart cities. *Computers and Electrical Engineering*, 105:108543, 2023.

[128] Arshin Rezazadeh, Davood Abednezhad, and Hanan Lutfiyya. MiGrror: Mitigating downtime in mobile edge computing, an extension to live migration. *Procedia Computer Science*, 203:41–50, 2022.

[129] Arshin Rezazadeh, Davood Abednezhad, and Hanan Lutfiyya. Hybrid-MiGrror: An extension to the hybrid live migration to support mobility in edge computing. *Journal of Ubiquitous Systems & Pervasive Networks*, 18(1):39–48, 2023.

[130] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[131] Roberto Rodrigues Filho, Renato S Dias, João Seródio, Barry Porter, Fábio M Costa, Edson Borin, and Luiz F Bittencourt. A self-distributing system framework for the computing continuum. In *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–10. IEEE, 2023.

[132] Swastika Roy, Hatim Chergui, and Christos Verikoukis. Towards bridging the FL performance-explainability trade-off: A trustworthy 6G RAN slicing use-case. *IEEE Transactions on Vehicular Technology*, 2024.

[133] Mamoon M Saeed, Rashid A Saeed, Maha Abdelhaq, Raed Alsaqour, Mohammad Kamrul Hasan, and Rania A Mokhtar. Anomaly detection in 6G networks using machine learning methods. *Electronics*, 12(15):3300, 2023.

[134] Niloy Saha, Mohammad Zangooei, Morteza Golkarifard, and Raouf Boutaba. Deep reinforcement learning approaches to network slice scaling and placement: A survey. *IEEE Communications Magazine*, 61(2):82–87, 2023.

[135] Muhammad Mohtasim Sajjad, Carlos J Bernardos, Dhammika Jayalath, and Yu-Chu Tian. Inter-slice mobility management in 5G: motivations, standard principles, challenges, and research directions. *IEEE Communications Standards Magazine*, 6(1):93–100, 2022.

[136] Luiz Carlos Silva de Sales. Algoritmo de seleção de políticas de migração de máquinas virtuais em função de diferentes demandas de recursos. 2023.

[137] Eduardo Felipe Zambom Santana, Ana Paula Chaves, Marco Aurelio Gerosa, Fabio Kon, and Dejan S Milojicic. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Computing Surveys (Csur)*, 50(6):1–37, 2017.

[138] Annisa Sarah, Gianfranco Nencioni, and Md Muhidul I Khan. Resource allocation in multi-access edge computing for 5G-and-beyond networks. *Computer Networks*, 227:109720, 2023.

[139] M. Scarpiniti, E. Baccarelli, and A. Momenzadeh. VirtFogSim: A parallel toolbox for dynamic energy-delay performance testing and optimization of 5G mobile-fog-cloud virtualized platforms. *MDPI Applied Sciences*, 9(6), March 2019.

[140] Kah Phooi Seng, Paik Jen Lee, and Li Minn Ang. Embedded intelligence on FPGA: Survey, applications and challenges. *Electronics*, 10(8):895, 2021.

[141] M Series. IMT vision–framework and overall objectives of the future development of imt for 2020 and beyond. *Recommendation ITU*, pages 2083–0, 2015.

[142] Syed Danial Ali Shah, Mark A Gregory, and Shuo Li. Toward network slicing enabled edge computing: A cloud-native approach for slice mobility. *IEEE Internet of Things Journal*, 2024.

[143] Koosha Sharifani and Mahyar Amini. Machine learning and deep learning: A review of methods and applications. *World Information Technology and Engineering Journal*, 10(07):3897–3904, 2023.

[144] Sparsh Sharma and Ajay Kaul. VANETs cloud: architecture, applications, challenges, and issues. *Archives of Computational Methods in Engineering*, 28:2081–2102, 2021.

[145] Yandong Shi, Lixiang Lian, Yuanming Shi, Zixin Wang, Yong Zhou, Liqun Fu, Lin Bai, Jun Zhang, and Wei Zhang. Machine learning for large-scale optimization in 6G wireless networks. *IEEE Communications Surveys & Tutorials*, 2023.

[146] Manoel C Silva Filho, Raysa L Oliveira, Claudio C Monteiro, Pedro RM Inácio, and Mário M Freire. CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE symposium on integrated network and service management (IM)*, pages 400–406. IEEE, 2017.

[147] Ahmed Slalmi, Hasna Chaibi, Abdellah Chehri, Rachid Saadane, and Gwanggil Jeon. Toward 6G: Understanding network requirements and key performance indicators. *Transactions on Emerging Telecommunications Technologies*, 32(3):e4201, 2021.

[148] SONATA. D6.3 final demonstration, roadmap and validation results. `https://project.sonata-nfv.eu/content/d63-final-demonstration-roadmap-and-validation-results`, 2018.

[149] C. Sonmez, A. Ozgovde, and C. Ersoy. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11), Aug. 2018.

[150] Paulo S Souza, Tiago Ferreto, and Rodrigo N Calheiros. Edgesimpy: Python-based modeling and simulation of edge computing resource management policies. *Future Generation Computer Systems*, 148:446–459, 2023.

[151] Charles Ssengonzi, Okuthe P Kogeda, and Thomas O Olwal. A survey of deep reinforcement learning application in 5G and beyond network slicing and virtualization. *Array*, 14:100142, 2022.

[152] Xiao Su, Jianpeng Qi, Jiahao Wang, Rui Wang, and Yuan Yao. EasiEI: A simulator to flexibly modeling complex edge computing environments. *IEEE Internet of Things Journal*, 11(1):1558–1571, 2023.

[153] Fengxiao Tang, Bomin Mao, Yuichi Kawamoto, and Nei Kato. Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption. *IEEE Communications Surveys & Tutorials*, 23(3):1578–1598, 2021.

[154] Muhammad Ashar Tariq, Malik Muhammad Saad, Mahnoor Ajmal, Ayesha Siddiqa, Junho Seo, Yang Haishan, and Dongkyun Kim. Network slice traffic demand prediction for slice mobility management. In *2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 281–285. IEEE, 2024.

[155] Belma Turkovic, Sjors Nijhuis, and Fernando Kuipers. Elastic slicing in programmable networks. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 115–123. IEEE, 2021.

[156] Zaib Ullah, Fadi Al-Turjman, Leonardo Mostarda, and Roberto Gagliardi. Applications of artificial intelligence and machine learning in smart cities. *Computer Communications*, 154:313–323, 2020.

[157] Karima Velasquez, David Perez Abreu, Marilia Curado, and Edmundo Monteiro. Resource orchestration in 5G and beyond: Challenges and opportunities. *Computer Communications*, 192:311–315, 2022.

[158] Kanupriya Verma, Ashok Kumar, Mir Salim Ul Islam, Tulika Kanwar, and Megha Bhushan. Rank based mobility-aware scheduling in fog computing. *Informatics in Medicine Unlocked*, 24:100619, 2021.

[159] Cheng-Xiang Wang, Xiaohu You, Xiqi Gao, Xiuming Zhu, Zixin Li, Chuan Zhang, Haiming Wang, Yongming Huang, Yunfei Chen, Harald Haas, et al. On the road to 6G: Visions, requirements, key technologies, and testbeds. *IEEE Communications Surveys & Tutorials*, 25(2):905–974, 2023.

[160] Gang Wang, Gang Feng, Tony QS Quek, Shuang Qin, Ruihan Wen, and Wei Tan. Reconfiguration in network slicing—optimizing the profit and performance. *IEEE Transactions on Network and Service Management*, 16(2):591–605, 2019.

[161] Haozhe Wang, Yulei Wu, Geyong Min, and Wang Miao. A graph neural network-based digital twin for network slicing management. *IEEE Transactions on Industrial Informatics*, 18(2):1367–1376, 2022.

[162] Theresa Wettig and Zoltán Adám Mann. Simulation-based analysis of threats to location privacy in fog computing. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 736–741. IEEE, 2021.

[163] S. Wijethilaka and M Liyanage. Survey on network slicing for internet of things realization in 5G networks. *IEEE Communications Surveys & Tutorials*, 2021.

[164] Guoquan Wu, Bing Zhang, and Ya Li. Intelligent and survivable resource slicing for 6G-oriented UAV-assisted edge computing networks. *Computer Communications*, 202:154–165, 2023.

[165] Yulei Wu, Hong-Ning Dai, Haozhe Wang, Zehui Xiong, and Song Guo. A survey of intelligent network slicing management for industrial IoT: Integrated approaches for smart transportation, smart energy, and smart factory. *IEEE Communications Surveys & Tutorials*, 24(2):1175–1211, 2022.

[166] Girma M Yilma, Zarrar F Yousaf, Vincenzo Sciancalepore, and Xavier Costa-Perez. Benchmarking open source NFV MANO systems: OSM and ONAP. *Computer communications*, 161:86–98, 2020.

[167] Hao Yu, Zhao Ming, Chenyang Wang, and Tarik Taleb. Network slice mobility for 6G networks by exploiting user and network prediction. In *ICC 2023-IEEE International Conference on Communications*, pages 4905–4911. IEEE, 2023.

[168] Tingting Yuan, Wilson Da Rocha Neto, Christian Esteve Rothenberg, Katia Obraczka, Chadi Barakat, and Thierry Turletti. Machine learning for next-generation intelligent transportation systems: A survey. *Transactions on emerging telecommunications technologies*, 33(4):e4427, 2022.

[169] Amr M Zaki, Sara A Elsayed, Khalid Elgazzar, and Hossam S Hassanein. Heuristic-based proactive service migration induced by dynamic computation load in edge computing. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 5668–5673. IEEE, 2022.

[170] Amr M Zaki and Sameh Sorour. Proactive migration for dynamic computation load in edge computing. In *ICC 2022-IEEE International Conference on Communications*, pages 4275–4280. IEEE, 2022.

[171] Sardar Khaliq uz Zaman, Ali Imran Jehangiri, Tahir Maqsood, Nuhman ul Haq, Arif Iqbal Umar, Junaid Shuja, Zulfiqar Ahmad, Imed Ben Dhaou, and Mohammed F Alsharekh. LiMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing. *Cluster Computing*, 26(1):99–117, 2023.

[172] Sardar Khaliq Uz Zaman, Ali Imran Jehangiri, Tahir Maqsood, Arif Iqbal Umar, Muhammad Amir Khan, Noor Zaman Jhanjhi, Mohammad Shorfuzzaman, and Mehedi Masud. COME-UP: Computation offloading in mobile edge computing with LSTM based user direction prediction. *Applied Sciences*, 12(7):3312, 2022.

[173] Sardar Khaliq uz Zaman, Tahir Maqsood, Faisal Rehman, Saad Mustafa, Muhammad Amir Khan, Neelam Gohar, Abeer D Algarni, and Hela Elmannai. Content caching in mobile edge computing based on user location and preferences using cosine similarity and collaborative filtering. *Electronics*, 12(2):284, 2023.

[174] Lanfranco Zanzi. *Machine learning-based orchestration solutions for future slicing-enabled mobile networks*. PhD thesis, Technische Universität Kaiserslautern, 2022.

[175] Xuanhong Zhou, Muhammad Bilal, Ruihan Dou, Joel JPC Rodrigues, Qingzhan Zhao, Jianguo Dai, and Xiaolong Xu. Edge computation offloading with content caching in 6G-enabled IoV. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[176] Mohammad Zolghadri, Parvaneh Asghari, Seyed Ebrahim Dashti, and Alireza Hedayati. Resource allocation in fog–cloud environments: State of the art. *Journal of Network and Computer Applications*, 227:103891, 2024.