UNIVERSIDADE D
COIMBRA

Miguel Alves Pimenta

5G Network Slicing

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering and Communications, Services and Infrastructure, advised by Professor Dr. Bruno Miguel de Oliveira Sousa, co-advised by Dr. David Abreu and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Miguel Alves Pimenta

5G Network Slicing

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering and Communications, Services, and Infrastructure, advised by Professor Dr. Bruno Miguel de Oliveira Sousa, co-advised by Dr. David Abreu and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2023

Acknowledgements

I would like to express my heartfelt gratitude to Professor Dr. Bruno Miguel de Oliveira Sousa for his unwavering support, guidance, and invaluable insights throughout the entire journey of this thesis. His dedication to excellence, passion for research, and commitment to fostering intellectual growth have been instrumental in shaping this work.

I am equally grateful to Dr. David Abreu and Noé Godinho, who served as co-advisors for this thesis. Their expertise, constructive feedback, and encouragement significantly contributed to the development and refinement of my research. Their diverse perspectives and collaborative spirit have broadened my horizons and enhanced the overall quality of this work.

To my family, I owe an immeasurable debt of gratitude. Your unwavering support, patience, and belief in my abilities have been my constant source of strength. You have been my pillar of strength, and I am profoundly thankful for your love and encouragement.

I would also like to extend my appreciation to my friends who have stood by me, providing emotional support and inspiration during both the challenging and joyful moments of this academic journey. Your camaraderie has made this experience more fulfilling and memorable.

To all the individuals, colleagues, and mentors who have played a part, big or small, in shaping this thesis, I extend my sincere thanks. Your collective contributions have enriched this work and my personal growth as a researcher.

Finally, I express my gratitude to the academic institution and all the resources it has provided, which made this research possible.

Thank you all for being a part of this important chapter in my academic and personal life.

Miguel Pimenta

Abstract

5G-beyond technology is seen as the future of telecommunications, with the ability to meet customer demands for high-quality service. In order to achieve this, the physical network infrastructure must be divided into multiple isolated logical networks, known as network slicing, each dedicated to different types of services based on their specific requirements. This approach allows for efficient and flexible use of network resources, enabling the support of a wide range of services such as Enhanced Mobile Broadband (eMBB), Ultra-reliable, low-latency communications (URLLC), and Massive Machine-Type Communications (mMTC) on a single network infrastructure.

Our goal is to optimize network slicing by implementing a model to make better use of available resources and to enhance the overall performance of network slicing. To achieve this, we have developed a comprehensive framework that enables the simulation of various types of networks. This framework allows users to specify the network details, such as the number of slices, services, nodes, and link attributes, and utilizes an optimization model to find the best solution for allocating services in each slice of the network in a fair manner.

Keywords

5G, Network Slicing, Fairness, Resource Allocation, Network Function Virtualization (NFV), Software Defined Networking (SDN).

Contents

Acronyms

2D Two Dimention.

3D Three Dimention.

4G Fourth-Generation.

5G Fifth-Generation.

AAI Active and Available Inventory.

AI Artificial Intelligence.

AMF Core Access and Mobility Management Function.

API Application Programming Interface.

APP-C Application Controller.

BA Barabási-Albert.

BPMN Business Process Model and Notation.

CIs Critical Infrastructures.

CMM Centralized Mobility Management.

CN 5G Core Network.

COTS Commercial off the Shelf.

CRD Custom Resource Definition.

DB Database.

DCAE Data Collection, Analytics, and Events.

DMM Distributed Mobility Management.

DN Data Network.

DoS Denial of Service.

E2E End-to-End.

eMBB Enhanced Mobile Broadband.

gNB Next Generation NodeB.

IaaS Infrastructure as a service.

ICT Information and Communication Technology.

InPs Infrastructure Providers.

Inter-RAT Inter-Radio Access Techonology mobility.

Intra-RAT Intra-Radio Access Techonology mobility.

IoT Internet of Things.

IPTV (Internet Protocol television.

ITSs Intelligent Transportation Systems.

LTE Long Term Evolution.

MANO Management and Orchestration.

M-CORD Mobile Central Office Re-architected as a Datacenter.

MCS Mission Critical Services.

MEC Multi-Access Edge Computing.

MIPS Million Instructions per Second.

MME Mobility Management Entity.

mMTC Massive Machine-Type Communications.

NFV Network Function Virtualization.

NFVI Network Function Virtualization Infrastructure.

NFVI-PoP Network Function Virtualization Infrastructure - Point of Presence.

NG-RAN New Generation Radio Access Network.

NIST National Institute of Standards and Technology.

NoSQL Not Only SQL.

NumPy Numerical Python.

OAI-CN OpenAirInterface Core Network.

OAI-RAN OpenAirInterface Radio Access Network.

ODL OpenDayLight.

ONAP Open Networking Automation Platform.

OOF ONAP Optimization Framework.

OOM ONAP Operations Manager.

OREOS Orchestration and Resource optimization for Reliable and low-latency Services.

OSS Operations Support Systems.

PaaS Platform as a service.

PCF Policy Control Function.

PDU Protocol Data Unit.

QoE Quality of Experience.

QoS Quality of Service.

QoS Flow Quality of Service Flows.

RAN Radio Access Network.

RANs Radio Access Networks.

rest API Representational State Transfer Application Programming Interface.

RESTful Representational State Transfer.

SaaS Software as a Service.

SDC Service Design and Creation.

SDK Software Development Kit.

SDN Software Defined Networking.

SDN-C Software-Defined Networking Controller.

SMF Session Management Function.

SO Service Orchestration.

SOAP API Simple Object Access Protocol Application Programming Interface.

SPs Service Providers.

TOSCA Topology and Orchestration Specification for Cloud Applications.

UDM Unified Data Management.

UPF User Plane Function.

URLLC Ultra-reliable, low-latency communications.

VFC Virtual Function Controller.

VIMs Virtual Infrastructure Managers.

VM Virtual Machine.

WMNs Wireless Mesh Networks.

List of Figures

Chapter 1

Introduction

5G Network, also known as the fifth generation of cellular mobile communications, is a highly advanced technology that offers a range of benefits over previous generations of mobile networks. One of the critical advantages of 5G is its faster speed, which enables a wide range of new and innovative use cases and services. Additionally, 5G networks offer lower latency, which is critical for applications that require real-time responsiveness, such as self-driving cars. Finally, 5G networks provide increased capacity, allowing them to support a large number of devices and high traffic volumes, making them suitable for use cases such as smart cities and Internet of Things (IoT) applications.

5G network slicing is made possible by the advanced capabilities of 5G networks. This allows operators to create multiple virtual networks on a single physical infrastructure and allocate resources flexibly and efficiently. This concept enables operators to efficiently allocate network resources and configure their services to meet the needs of their customers. The allocation of resources and configuration of the slices are made possible by orchestrators, which offer a reliable and efficient functioning of network slices.

Orchestrators can be physical devices or software programs that use algorithms and other tools to automate the process of slice creation and management. With a comprehensive view of the network architecture, orchestrators can effectively manage the network, increasing its efficiency while reducing the cost and complexity of managing multiple virtual networks.

1.1 Objectives

This thesis is carefully designed to meet its goals by thoroughly investigating both traditional and modern solutions for network slicing using Software Defined Networking (SDN).

The thesis takes a deep dive into various strategies used in network slicing. A key focus is the validation of a fairness model tailored to enhance resource allocation within network slicing. This model emphasizes fairness, aiming to ensure that

each network slice gets a fair share of resources like bandwidth and CPU. The importance of fairness is highlighted as we're committed to making sure every slice gets the right amount of resources without any imbalance.

It's important to clarify that the focus of this thesis does not lie in creating the fairness model. The structure and formulation of this model have already been established by Noé Godinho. Initially conceptualized with the main aim of furnishing a proficient resource allocation solution in network contexts, this model was developed as part of the project Orchestration and Resource optimization for Reliable and low-latency Services (OREOS).

The objective at hand is to assess and affirm the suitability of this model for integration into the current project, which calls for a resource allocation algorithm. This validation process aims to confirm the applicability and effectiveness of the existing model within the specific context of this project.

1.2 Contributions

This work contributes to a novel network resource allocation framework, specifically tailored to address the challenges of the OREOS project. These challenges center around the necessity for a resource allocation model that optimally distributes resources within network slicing.

The objective of the OREOS project is to design and implement an end-to-end orchestration platform for provisioning and managing critical services, such as vehicle communications, electrical distribution networks, and emergency communications for public safety [OREOS: Estudos Preliminares na área do Projeto].

The framework presented in this thesis provides a powerful tool to simulate diverse network scenarios, optimizing resource allocation while minimizing costs and adhering to specific constraints. The Fairness model presented in this framework provides the OREOS project valuable insights into efficient resource allocation, enhanced service provisioning, and ensuring fairness among network slices. The validation process confirms the framework's efficiency and reliability, establishing its applicability to real-world network environments.

Through integration into the OREOS project, our developed framework enables the effective orchestration of critical services (such as vehicle communications, electrical distribution networks, or emergency communications for public safety). By leveraging the capabilities of the fairness model, the project can achieve improved performance, reliability, and low-latency communication, making significant advancements in critical services management and provisioning.

Another significant contribution of this thesis lies in the creation of a scientific article that provides an exhaustive exposition of the 5G network, elucidating its intricacies and operational dynamics. This article will serve as a comprehensive guide aimed at rendering the complexities of 5G comprehensible to a broader audience. Additionally, it will demonstrate the practical application of the network resource allocation framework developed within the framework of network slic-

ing. This demonstration will be conducted in a manner that is transparent and straightforward, ensuring that the essential concepts and methodologies are accessible to both experts and newcomers in the field.

1.3 Structure

This document is organized as follows:

- **Chapter 2 -** This chapter provides an overview of key topics such as Network Function Virtualization (NFV), Multi-Access Edge Computing (MEC), and network slicing. It also reviews graph analysis and optimization tools, orchestration platforms, and the concept of fairness in resource allocation in network slices.
- **Chapter 3 -** This chapter reviews related work related to models for Fairness and several works for 5G.
- **Chapter 4 -** This chapter describes the research objectives of the work and introduces the approach and methodology used throughout the dissertation.
- **Chapter 5 -** This chapter explains our Fairness Model.
- **Chapter 6 -** This chapter delves into our framework. How it works, which components are used in its creation and why.
- **Chapter 7 -** This chapter refers to the process of validation of the fairness model. The metrics used and the test cases that we conducted.
- **Chapter 8 -** Provides the main conclusions and the future work that will follow.

Chapter 2

Background and Terminology

This chapter provides an introduction to the latest network communication technology, the 5G network. The functionalities and characteristics of 5G technology are detailed, along with its applications in modern network methods. Understanding these concepts is essential for comprehending the topics covered in this thesis.

The chapter begins by introducing some of the key technologies used in network slicing. Each technology is briefly defined, along with its contributions to network slicing. These technologies play a crucial role in enabling the implementation of network slicing and optimizing resource allocation.

Network slicing is a fundamental concept explored in this section. The term "network slicing" is defined, and its significance in the context of 5G networks is explained. Network slicing allows network operators to partition the infrastructure into multiple virtual networks, each tailored to specific services or applications. This flexibility in network customization is a critical feature of 5G networks.

To implement network slicing effectively, specific components are essential. This chapter delves into the components required for network slicing, such as Network Function Virtualization (NFV) and Software Defined Networking (SDN). These components enable dynamic resource allocation and service isolation, contributing to the success of network slicing in 5G environments.

In addition to the technical aspects of network slicing, this chapter also addresses the concept of fairness. Fairness in resource allocation is a vital consideration in multi-slice networks, as it ensures that all services and users receive equitable access to network resources. The chapter discusses how fairness is achieved in network slicing and the role it plays in optimizing overall network performance.

Finally, the chapter introduces two important orchestrators, Open Networking Automation Platform (ONAP) and MOSAIC5G, which play a key role in network slicing. An overview of their components and functionalities is provided. Orchestrators act as central management systems that coordinate and automate various network operations, including resource allocation, service deployment, and network configuration. Their importance in network slicing lies in their ability to efficiently manage the diverse and dynamic requirements of multiple net-

work slices.

2.1 Basic Concepts

It is vital to comprehend a few principles before discussing the 5G network and network slicing concepts. In this section, a variety of technologies are presented to give the reader a few notions on how these technologies are essential to implementing 5G network slicing. The principles that follow in this section were written with the aid of a survey about 5G network slicing [Barakabitze et al., 2020], which thoroughly clarified them.

2.1.1 Software defined Networking

SDN aims to increase the flexibility and agility of networks. SDN is an approach to plan, deploy, and administer networks, that separate the control and data/forwarding planes. This separation results in flexibility and centralized control with a global view of the entire network, along with the capability to respond rapidly to changing network conditions, business, market, and end-user needs. As shown in the image below, SDN bridges the gap between service provisioning and network management by establishing a virtualized control plane. This virtualized control plane, executes intelligent management decisions among network operations, as illustrated in figure 2.1.

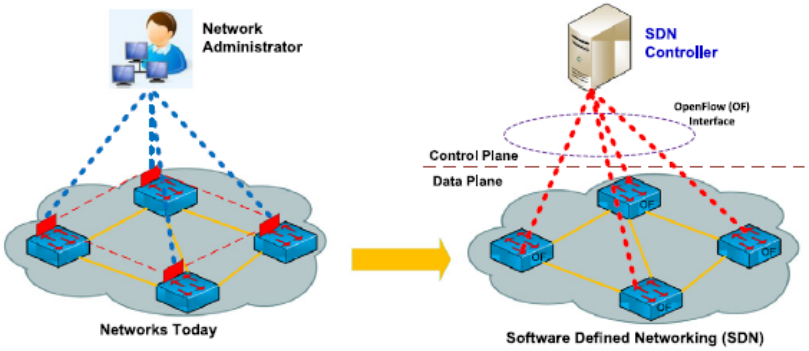


Figure 2.1: SDN implementation[Barakabitze et al., 2020]

Some of the SDN solutions that can support network slicing in 5G systems are known as OpenDayLight (ODL) [ODL] and Mobile Central Office Re-architected as a Datacenter (M-CORD) [M-CORD].

ODL can optimize softwarized and virtualized networks, to provide dynamic services.

M-CORD uses SDN and NFV to offer services to operators in the context of setting up 5G mobile wireless networks. The end-to-end slicing provided by M-CORD includes the option of a virtualized and programmable Radio Access Network (RAN).

2.1.2 Network Function Virtualization

NFV corresponds to a software implementation of a dedicated function, normally implemented in hardware like routers, firewalls, and load balancers. A NFV runs on a virtual machine, using the physical hardware's available processor, memory, and storage as if it were its computer.

The NFV paradigm decouples software from the hardware platform, allowing independence between NFVs. The NFV paradigm offers more freedom in the implementation of network services because software components are not coupled to hardware and can carry out a variety of tasks at different times. Network operators can therefore launch innovative or new services while continuing to use the same hardware platform. Due to this decoupling feature, NFV offers dynamic network operation and service delivery since network operators can provide customized services while dynamically scaling NFV performance by customer preferences. In addition, network resources can be efficiently allocated to NFVs through dynamic scaling.

This feature of decoupling software from hardware, results in low latency and low failure rates for NFV operations, as well as optimized resource provisioning to end users with high Quality of Service (QoS).

2.1.3 Cloud Computing

According to the National Institute of Standards and Technology (NIST), cloud computing is a "model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [Mell et al., 2011]

Cloud computing allows for the ability to scale up or down according to the business needs, which helps to reduce operational expenses. The pay-per-use model for providing resources eliminates the need for costly hardware investments, as users only pay for the workloads and resources that they use. This detailed measurement of compute resources ensures that businesses can adapt to changing demands and lower their operational expenses by only paying for what they use.

In a cloud computing environment, two categories build the typical service provider roles:

- **Infrastructure Providers (InPs):** Handle cloud platforms and resource leasing with the use of a pricing scheme.
- **Service Providers (SPs):** To serve end users, one or more InPs may lease resources.

The services that cloud computing offers, are divided into three main categories or types of cloud computing:

- **Software as a Service (SaaS):** Is a method of distributing programs (web services) via the Internet. Users can use a computer or mobile device with internet connectivity to access SaaS applications and services from any location.

- **Platform as a service (PaaS):** It offers a platform that enables users to create, use, and manage different apps without having to deal with the complexity of setting up and managing cloud infrastructure.

- **Infrastructure as a service (IaaS):** It offers self-service approaches for administering, accessing, and managing the network, computing, and storage services found in remote data centers. For example, Amazon Web Services supply a virtual server instance and storage, as well as Application Programming Interface (API) that lets users migrate workloads to a Virtual Machine (VM). Users have an allocated storage capacity and can start, stop, access and configure the VM and storage as desired.

We may better understand these principles by using the practical examples in figure 2.2.

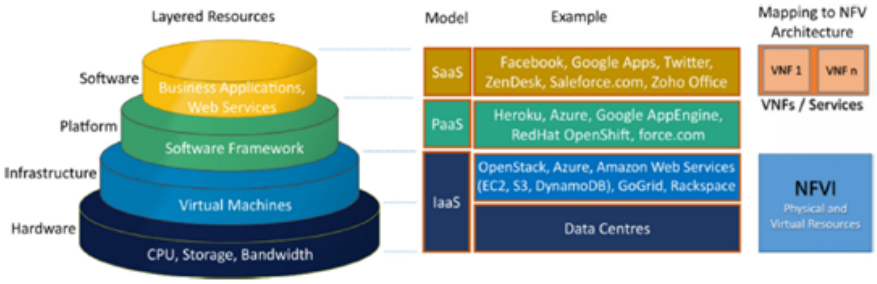


Figure 2.2: Cloud Computing Services categorized[Barakabitze et al., 2020]

2.1.4 Multi-access edge computing

Multi-Access Edge Computing (MEC) relocates traffic and service processing from a centralized cloud to the network's edge, bringing it closer to the user. The network edge processes, stores, and analyzes data instead of sending it all to a cloud for processing, allowing high-bandwidth applications to operate in real-time by reducing latency and bringing data collection and processing closer to the client [MEC].

The MEC platform, and MEC apps share communication interfaces through MEC services which function as a middleman API between them, and also can provide services to the applications. MEC Services nodes can operate locally in the deployed data center or remotely in the cloud.

2.2 Graph Analysis and Optimization Tools

In this section, we explore the essential graph analysis and optimization tools that form the foundation of our resource allocation framework for network slicing. These tools enable us to model, analyze, and optimize the allocation of resources in multi-slice networks. We introduce NetworkX, a versatile Python package for graph analysis, MiniZinc, a powerful constraint modeling language, and Matplotlib, a plotting library that aids in visualizing resource allocation outcomes.

2.2.1 NetworkX: Comprehensive Network Analysis

"NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks" [NetworkX].

Its comprehensive set of tools and functionalities make it an invaluable resource for researchers working with network data. With a focus on efficiency, versatility, and seamless integration with the Python ecosystem, NetworkX serves as a powerful graph analysis library that has played a fundamental role in the development of our resource allocation framework for network slicing.

One of the key strengths of NetworkX lies in its support for various graph data structures, including directed graphs (DiGraphs), undirected graphs, and multigraphs. These structures efficiently represent the intricate relationships between nodes and edges in real-world networks, enabling us to model complex systems with ease.

To delve more profoundly into NetworkX's capabilities and its potential as an enriching tool for network simulation, our exploration draws from a range of scholarly works, including references like [5G Dataset Network Slicing CRAWDAD Shared], [Nerini, 2020], and [Resilient service chains through smart replication]. Through these sources, we gain deeper insights into the tool's functionalities and its significance within the context of network analysis and simulation.

2.2.2 MiniZinc: Modeling Resource Allocation

MiniZinc is a versatile constraint modeling language that facilitates mathematical modeling and the solution of combinatorial optimization problems. By serving as an intermediary layer between users and solvers, MiniZinc allows for a focus on problem logic rather than solver implementation details.

MiniZinc employs a declarative syntax, enabling concise expression of optimization problems without specifying solution algorithms explicitly. This approach enhances problem-solving simplicity and model readability.

At its core, MiniZinc adopts constraint-based modeling, employing constraints to define variable relationships and limit their potential values. This flexibility enables the expression of problem-specific rules, preferences, and logical condi-

tions. With support for various constraint types, such as arithmetic, logical, and global constraints, MiniZinc offers an expressive framework for diverse combinatorial problem formulations.

A prominent advantage of MiniZinc is its solver independence, allowing users to model a problem once and then utilize multiple solvers for solution finding, such as Choco and Gecode which are used in constraint programming. This adaptability facilitates effortless experimentation and comparison of various solver strategies and algorithms, integrating a range of state-of-the-art solvers for tackling diverse optimization challenges.

Gecode is renowned for its outstanding efficiency in solving combinatorial optimization and constraint satisfaction problems. Given the complex nature of our fairness model, involving various constraints and objectives, a solver with expertise in constraint programming was essential, and Gecode fit the bill perfectly.

2.2.3 Matplotlib: Visualizing Resource Allocation

Matplotlib is a highly popular Python library used for creating static, interactive, and animated data visualizations. It offers a wide range of Two Dimention (2D) and Three Dimention (3D) plots and is widely regarded as one of the top plotting libraries in the Python ecosystem. The best part about Matplotlib is its user-friendly interface, making it accessible to both beginners and experienced developers. You can generate plots with just a few lines of code, but it also allows advanced users to customize their visualizations extensively.

With Matplotlib, you can create various plot types such as line plots, scatter plots, bar plots, histograms, pie charts, and 3D plots, among many others. This versatility makes it suitable for visualizing different types of data.

One of the great things about Matplotlib is its seamless integration with Numerical Python (NumPy), the fundamental package for scientific computing in Python. This integration lets you plot data directly from NumPy arrays and perform mathematical operations with ease.

Matplotlib plays a vital role in data analysis, exploratory data visualization, and communicating insights in various domains, including data science, engineering, finance, and academia. Its ease of use, wide adoption, and ability to create visually appealing data visualizations have made it an essential tool for Python programmers.

2.3 5G Network

The construction of Fifth-Generation (5G) mobile and wireless communication systems has been sparked globally by the expanding use of mobile video services (such as YouTube and mobile TV) on smart devices and developments in the Internet of Things. This section will address this subject by identifying the

key features that the 5G Network has to offer customers that set it apart from its predecessors [Barakabitze et al., 2020].

2.3.1 An introduction to 5G

The most recent technology, known as 5G, is designed to significantly speed up and improve the responsiveness of wireless networks making businesses market more efficient and giving consumers access to more information faster than ever before.

This kind of network can be applied to a variety of commercial settings where services must be delivered reliably, including smart grids, public safety, water delivery, and natural gas networks. Every 5G application has a unique set of performance requirements. Taking this into consideration, 5G Network has a variety of quality and business requirements, based on the article [Afolabi et al., 2018].

Data Rate and lower Latency

The 5G network is expected to provide 1–10 Gigabytes per second data rates, which are almost ten times of Fourth-Generation (4G) Long Term Evolution (LTE) network's theoretical peak data rate of 150 Megabytes per second, which for applications like 3D gaming, for example, will be mandatory.

By utilizing terminals with Artificial Intelligence (AI) capabilities, 5G will be able to supply high-level services with guaranteed end-user service quality and mobile broadband without the chance of failing even in crowded areas (such as stadiums, cars, trains, concerts, or shopping malls). Another goal of 5G is to support connectivity available anytime and anywhere with a 1-millisecond round-trip latency. These values were taking into account the article [Barakabitze et al., 2020]

Enhanced service availability, security, and mobility

To facilitate fast communications for emergency and public safety situations, 5G must have a strong, reliable, and resilient network. As mentioned before, 5G will support communication that cannot fail, like public safety, and smart grids. Thus, we need an enhanced service availability with high-speed connections.

Mobility Management is necessary to make sure that the user can "reach" the network, for example, to inform them of incoming calls and messages or a user's ability to start a conversation with other users or services like Internet access.

Centralized Mobility Management (CMM) is the universal mobility management system that is frequently used in LTE. However, CMM has significant problems [Akkari and Dimitriou, 2020], particularly in crowded networks where a greater proportion of mobile users would lead to more frequent handover requests and, as a result, a greater proportion of signaling messages.

The newly introduced Distributed Mobility Management (DMM) proposals for 5G, have the potential to overcome the current mobility management limitations [Cominardi et al., 2017]. The key differentiator of DMM is a simple distribution

of mobility anchors by positioning a few of them closer to the user's location.

Finally, for mission-critical applications like smart grids, public safety, water distribution, and natural gas networks, the 5G network will need to ensure and can protect against security threats, such as, Denial of Service (DoS) in which resources remain unavailable causing a significant impact on organizations [Barakabitze et al., 2020].

Quality of Experience (QoE)-based service billing and pricing

Billing/charging policies by service providers must be based on the QoE off the service. For example, a premium (Internet Protocol television (IPTV) customer who pays more for a service expects a better service quality, as a result, it is important to develop appropriate QoE-based service billing and pricing methods that will directly impact the quality of business while also offering a QoE differentiator in future 5G networks.

Longer battery life, seamless user experience, and context-aware networking

Sensor networks based on 5G deployment can only be feasible if everyday operations support several years of significantly longer battery life and decreased energy usage for 5G devices. Future 5G networks should be able to maintain the quality of possible latency and data rate while delivering and providing a consistent user experience regardless of the user's location. The characteristics of 5G solutions should also allow the network to adjust to the demands of applications and connected smart devices.

2.3.2 4G Network vs 5G Network

As stated previously, the rising customer demand for high-quality services and the expanding consumption of multimedia services have led to a fundamental shift in how we manage networks in terms of abstraction, segregation, and mapping of forwarding, control, and management aspects of services.

Faster speeds, more bandwidth, and lower "latency" or lag time in communications between devices and servers, are the three main distinctions between 4G and 5G. The 5G network is expected to provide the fastest data rates in comparison with the 4G LTE network. 5G also delivers a latency time under 5 milliseconds, while its predecessors 4G can only deliver 60 milliseconds to 98 milliseconds [5G vs 4G: Learn the key differences between them].

Another feature that must be considered is security, 4G networks provide safety at the user level, such as data encryption and network-level security. 5G Network works at a more advanced level like business, delivery models, services, etc. This means that 5G needs to be able to offer multiple layers of security assuring that attacks like Denial of Service never happen.

However, what is known as "softwarization" is the most novel feature. Softwarization in 5G networks will allow the implementation of Network Slicing by giving the possibility to virtualize a piece of hardware. In other words, a spe-

cific functionality will, for instance, be executed by a virtual machine in software as opposed to hardware. To create, deploy, administer, design, and maintain network components and services, this functionality makes use of software programming enabling 5G Network to provide its users with services that are higher quality and more cost-effective.

In the 5G era, developers and operators may quickly create application-aware networks (Network keeps up-to-date records of the applications that connect to it, maximizing both its performance and that of the applications or systems it governs) and network-aware (Capacity to keep an eye on factors affecting the network, such as available bandwidth, network capacity, packet loss rate, and delay, allowing the application to modify its performance for better resources' usage) apps to meet their business objectives thanks to the strength of softwarization and virtualization technologies.

2.3.3 5G Network Architecture

Before discussing Network Slicing and what it entails, it is necessary to comprehend how a 5G network operates inside, what its components are, and why they are significant. 5G has 3 main components domains: New Generation Radio Access Network (NG-RAN), 5G Core Network (CN) and Packet Data Transport.

These 3 domains are what needs to be in place to support network slicing, as depicted in figure 2.3.

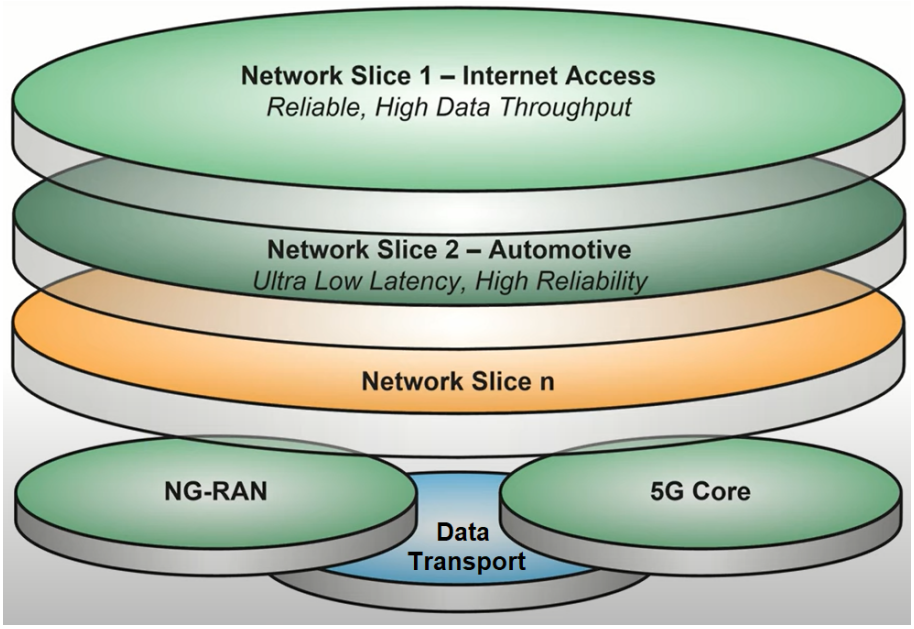


Figure 2.3: 5G Main Components [MPIRICAL 5G Core Network]

New Generation Radio Access Network

Modern technology integrates cutting-edge technological advances in 5G new radio air interface and 5G next-generation core network designs, making it substan-

tially more sophisticated than 4G networks. Fast, low-latency, high-bandwidth 5G Radio Access Networks (RANs) that can connect anything to anywhere with unmatched performance, efficiency, and cost are the end consequences, as we've seen before.

NG-RAN, is the newest wireless communication architecture that uses 5G radio frequencies to give devices wireless connectivity. A crucial element of a mobile communications system that uses radio links to link devices like smartphones to a network is the RAN. To do this, voice and data are converted into digital signals that are then transmitted as radio waves to RAN transceivers, who then forward them into the core network.

Core Network

In this sub-section, we present information about the composition of a core network, based on the educational videos provided by [MPIRICAL 5G Core Network].

The 5G Core Network manages all the internet and data connections. It is designed to integrate with the internet much more efficiently and manages the advanced features of 5G, like network function virtualization and network slicing.

To carry out network user traffic, 5G uses what is called a Protocol Data Unit (PDU) session, that runs from the device through the Next Generation NodeB (gNB), User Plane Function (UPF) and then on to the data network. This is shown in figure 2.4. In other terms, the connection between the device and the Data Network is made possible through PDU sessions.

gNB and UPF are two components of the 5G Core Network as we can see in figure 2.4, which we will discuss further in this section.

Based on what the service requires, PDU sessions can offer different degrees of service quality by using Quality of Service Flows (QoS Flow). There are numerous QoS flows in a PDU session, and each one has a distinct ID to set it apart from the others, once traffic can have varied QoS requirements. When no filters are required to be applied, a default QoS flow is always present in a PDU session, allowing all packets to pass through. Latency and priority are examples of QoS requirements.

As we can see in figure 2.4, there are various elements designed to keep that PDU session active for the subscriber and to ensure that the PDU session follows the subscriber's moves around the network.

Core Access and Mobility Management Function

As its name suggests, this component handles mobility management, allowing the Core Access and Mobility Management Function (AMF) to constantly be aware of the subscriber's location in a given traffic area.

The AMF also has a major role in security because it must communicate with other subscriber databases to confirm that the subscriber is authorized to use the

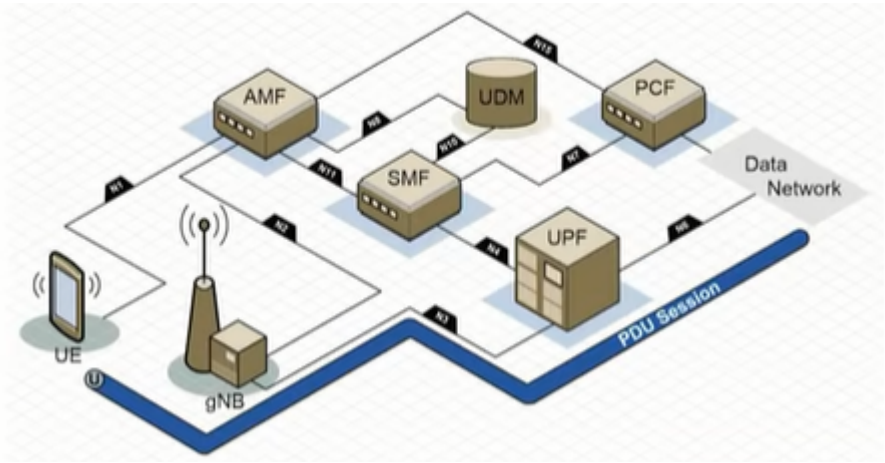


Figure 2.4: 5G Core Architecture[MPIRICAL 5G Core Network]

network.

Session Management Function

In LTE the Mobility Management Entity (MME) is responsible for the Session and Mobility Management, in 5G that functionality is being split so that the Mobility Management is left for the AMF and the Session Management is made in the Session Management Function (SMF).

SMF is responsible for communicating with the Policy Control Function (PCF) to determine whether or not a particular user data session is allowed to proceed in the network.

The actual PDU session connectivity, which involves setting up several different connections in the network, is another function that the SMF is concerned with. These connections go through the UPF, and the SMF is responsible for deciding which UPF to use.

User Plane Function

As we stated earlier a PDU Session runs from the device through the gNB, UPF, and then on to the data network.

The UPF is important in the data transfer process interconnecting the Data Network (DN) in the 5G architecture. Additionally, it handles packet inspections, and QoS handling, and acts as an anchor point for Inter-Radio Access Techonology mobility (Inter-RAT) and Intra-Radio Access Techonology mobility (Intra-RAT) mobility. UPF provides a high-performance forwarding engine for user traffic.

On the other hand, gNB enables devices wireless communication. It is in charge of controlling the wireless connectivity between devices and the 5G Network.

Unified Data Management

Unified Data Management (UDM) is a central database with subscriber data that focuses on three key areas:

- **Access Authorization:** Holds security keys.

- **Registration and Mobility Management:** Keeps track of the AMF to which our subscriber is allocated.
- **Data Network Profile(s):** Contains the subscriber's profile containing information about what the subscriber is allowed to do, which data networks it can connect and what kind of QoS profile he can expect to be granted when they connect to those data networks.

Policy Control Function

The PCF provides policy control for sessions' management, access and mobility, and PDU Session selection. The SMF and application functions work together with the PCF to offer permitted QoS for session management. When choosing a PDU Session, PCF also consults the SMF to see if any network issues could affect how the subscribers can use the services. The PCF communicates with the AMF to monitor service area restrictions and control access and mobility policies.

All of these elements come together to form the 5G Core, and a number of them can be virtualized as part of the NFV infrastructure. This indicates that instead of being used as standalone devices, these components are now running as software processes on Commercial off the Shelf (COTS) servers. Therefore, the concept behind Network Functions Virtualization is that we may create an NFV Infrastructure that offers these software processes the compute, storage, and network resources they will eventually need. The benefits of employing NFV include the fact that this infrastructure is based on COTS hardware, which significantly reduces the cost of deploying the Core Network, and the ease with which these processes can be scaled up or down because they can run as software processes. It requires a Management and Orchestration (MANO) to support an efficient management of NFVs.

2.4 Network Slicing

The industry has adopted 5G as the next-generation network that can handle vertical applications (pieces of software designed to fit the needs of a specific market, industry, or company) with a variety of service requirements. The physical network has to be sliced into multiple isolated logical networks of varying sizes and structures which are dedicated to different types of services based on their requirements, to implement this vision in 5G network [Barakabitze et al., 2020].

2.4.1 Network Slicing Introduction

A network slice is an End-to-End (E2E) logical network, running on a shared underlying (physical or virtual) infrastructure, that offers certain network capabilities and network characteristics to fulfill a given business goal of a customer.

When we create a network slice, that slice is logically separated from all other network slices and will support a set of attributes, qualities, and characteristics.

We can create multiple network slices, but does require very comprehensive slice orchestration. We need to be able to manage the network slices' life cycle management. Instantiate, maintain, and tear them down when they're no longer required. We need to independently monitor them.

As we can see in figure 2.5, we can have multiple slices in which one network slice is for mobile broadband and another slice is for connected vehicles. Mobile Broadband's slice in this scenario must have certain qualities, such as reliability and high data throughput, while the connected vehicle's slice must have extremely low latency and great reliability. Briefly, we can specify the characteristics we desire in each network slice to serve a specific function.

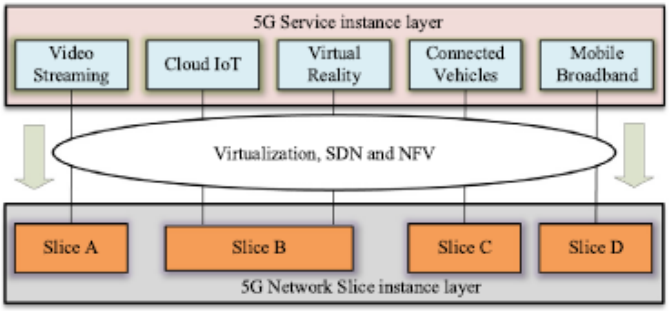


Figure 2.5: Cloud Computing Services categorized[Barakabitze et al., 2020]

In the context of 5G network slicing, three main categories have been defined:

- **Enhanced Mobile Broadband (eMBB):** This category focuses on delivering high-speed, high-capacity data connectivity to meet the escalating demands of data-intensive applications. eMBB aims to facilitate activities like high-definition video streaming, virtual and augmented reality experiences, and large file transfers, all requiring substantial data throughput and reliable connections. By allocating resources efficiently and optimizing data delivery, eMBB enhances user experiences and supports the growing appetite for bandwidth-intensive applications.

- **Massive Machine-Type Communications (mMTC)**: In contrast to eMBB's emphasis on high data rates, mMTC targets the pervasive connectivity of a massive number of devices, often seen in the Internet of Things (IoT) domain. This category accommodates scenarios where a multitude of devices, sensors, and objects need to communicate seamlessly, requiring efficient resource allocation, minimal energy consumption, and low data rates. mMTC lays the foundation for smart cities, industrial automation, and various sensor-driven ecosystems, transforming how devices interconnect and exchange data.

- **Ultra-reliable, low-latency communications (URLLC):** This category is tailored for applications demanding instantaneous response times and exceptional reliability, crucial for tasks such as real-time control, industrial automation, and mission-critical communication. URLLC ensures that latency is minimized to a fraction of a millisecond, and packet loss is virtually

nonexistent. This reliability is essential for applications where even the slightest delay or data loss can have significant consequences, underscoring URLLC's significance in fields such as autonomous driving, remote surgery, and advanced industrial processes.

.

2.4.2 Network Slicing Orchestration

As stated previously we need to be able to monitor all of the network slices life cycle management independently. Instantiate them, maintain them, and tear them down when they're no longer required. An orchestrator who has a broader understanding of the network is in charge of such functionality so that a Network Slice can be managed by its needs.

The orchestrator is required to realize the Network Slicing vision. It is responsible for automating the creation, monitoring, and implementation of resources and services in adjacent software and virtualized environments. The 5G architecture is quite complicated, as we have previously seen, and we require software that can understand the "big picture" of this architecture to deploy resources in specific components as necessary, depending on the resources that a particular slice needs.

Different types of orchestrators

The orchestrator provides orchestration of Network Function Virtualization Infrastructure (NFVI) across multiple Virtual Infrastructure Managers (VIMs) and lifecycle management of the network services, including instantiation, scale in/out (known as elastic scaling), performance measurements, event correlation, resource management, validation and authorization for resource requests, and policy management [Zhang and Meddahi, 2017].

Resource Orchestrator

The management and coordination of resources under the control of various VIMs is accomplished via resource orchestration. This kind of orchestrator manages NFVI resources within the Network Function Virtualization Infrastructure - Point of Presence (NFVI-PoP) or across other NFVI-PoPs (physical location where the infrastructure and resources needed to support the operation of virtualized network functions are deployed) by coordinating, authorizing, releasing, and engaging them.

Network Services Orchestrator

Network services orchestration manages and coordinates multiple network functions and services used to deliver the different slices. This type of orchestrator performs tasks such as allocating resources, such as bandwidth and processing power, to different slices and managing the communication between the slices and other network elements.

2.5 Orchestration platforms

This subsection provides an overview of diverse orchestration platforms, including ONAP and MOSAIC5G, which play a vital role in the administration of dynamic network slices and resource orchestration in 5G networks. These platforms are open-source and have been specifically developed to cater to the complexities of modern network environments.

For our work, we will focus solely on the orchestrators mentioned above, as the project Orchestration and Resource optimization for Reliable and low-latency Services (OREOS) is directly involved in the activities outlined in this dissertation. According to the project documentation, OREOS leverages the orchestration functionalities offered by ONAP and OSM, both of which are considered comprehensive solutions supporting various infrastructures, such as OpenStack and VMWare [OREOS: Estudos Preliminares na área do Projeto].

While ONAP is recognized as a highly complete orchestration solution, its complexity may pose resource challenges. Therefore, the study on MOSAIC5G aims to investigate how a less complex orchestrator can effectively orchestrate a network like ONAP. This analysis will provide valuable insights into different orchestrator platforms and their capabilities.

The information presented in this section is based on reputable sources, including articles and published materials available in references [Open Network Automation Platform] and [MOSAIC5G]. These references have been crucial in compiling the documentation for this research.

2.5.1 Open Networking Automation Platform

ONAP provides network operators, cloud providers, and businesses to orchestrate, manage, and automate network and edge computing services synchronously. The rapid automation of new services and comprehensive lifecycle management necessary for 5G and next-generation networks are made possible by real-time, policy-driven orchestration and automation of physical, virtual, and cloud-native network functions.

The ONAP project responds to the growing demand for a common automation platform that enables telecommunication, cable, and cloud service providers, as well as their solution providers, to deliver differentiated network services on demand, profitably, and aggressively, while leveraging existing investments.

Previous architectures, such as Operations Support Systems (OSS)/management software platforms required a longer cycle of software development and integration. Since ONAP is a cloud-native application composed of several services, requires a complex initial deployment as well as post-deployment management. The ONAP Operations Manager (OOM) manages and controls the full lifetime of ONAP components. In addition, OOM helps enhance ONAP platform maturity by providing scalability and resiliency enhancements to the components it

manages.

The ONAP platform enables a closed control loop approach that supports real-time response to responsive events, enabling end-user organizations and their network/cloud providers to cooperatively instantiate network components and services quickly and dynamically. There are three main prerequisites to develop, engineer, plan, bill, and ensure these dynamic services:

- A strong design framework that enables the specification of the service in all its aspects like modeling resources, and describing policy rules, among others.
- An orchestration and control framework (Service Orchestrator and Controllers) that follows recipes and policies to manage service demands elastically and enable automated service instantiation when necessary.
- An analytical framework that carefully examines the performance of the service throughout its life cycle based on the specified design, analytics, and policies

The ONAP architecture consists of Design-Time and Run-Time functions, as well as functions to manage ONAP itself. The design time framework is a full-featured development environment with tools, methods, and repositories for defining and specifying resources, services, and products. The rules, regulations, and other models distributed by the design and creation environment are carried out via the run-time execution framework. figure 2.6 gives us a clearer understanding of the ONAP components and how they are divided.

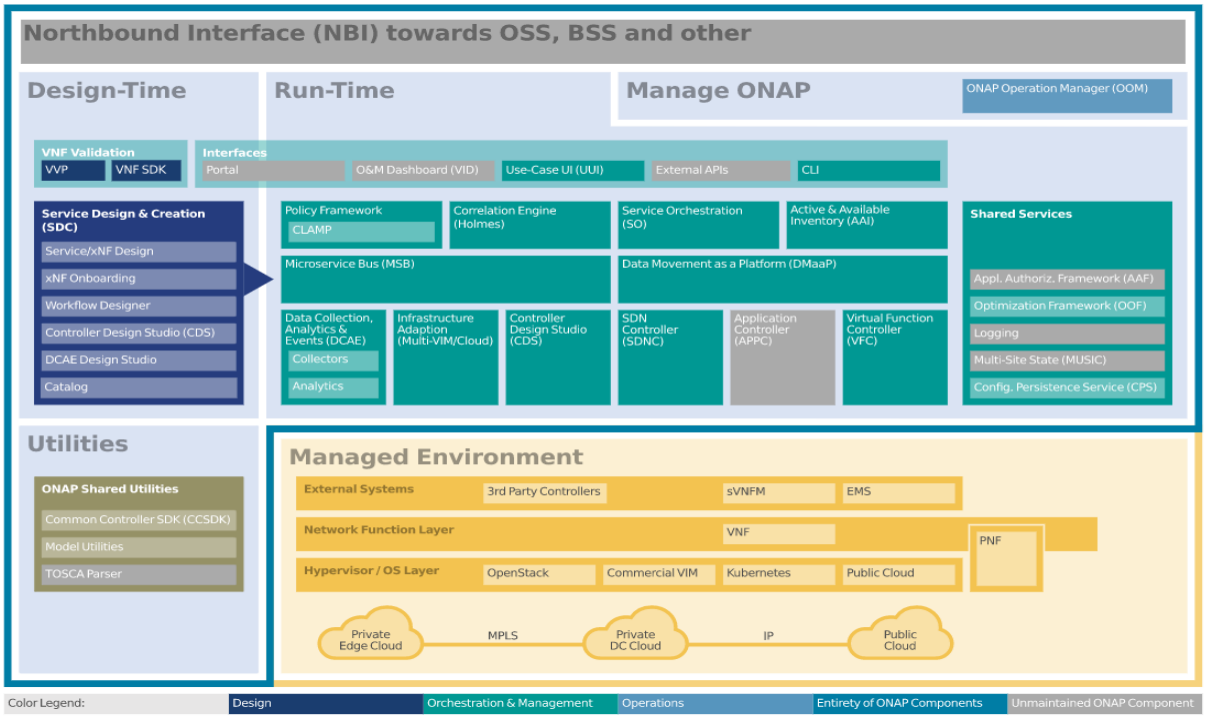


Figure 2.6: ONAP Overview [Open Network Automation Platform]

Service Design and Creation

Service Design and Creation (SDC) provides service design and creation functionality for an automation platform including:

- Resource onboarding: provides the ability to design NFV resources including NFVs, Services, and products.
- Service Design: creates all artifacts (models) that are required to instantiate and manage resources, services, and products on the ONAP platform.
- Launch resource and service testing.
- Deploy provided services into the network.
- Provides design time catalog capabilities.
- Process workflow support for talking the NFV or other resources and services through the process steps(design, test, deploy).

As shown in figure 2.7, SDC provides 3 functionally distinct layers with modular software, integrated with internal API. SDC offers a robust design time user experience to the service provider, by allowing:

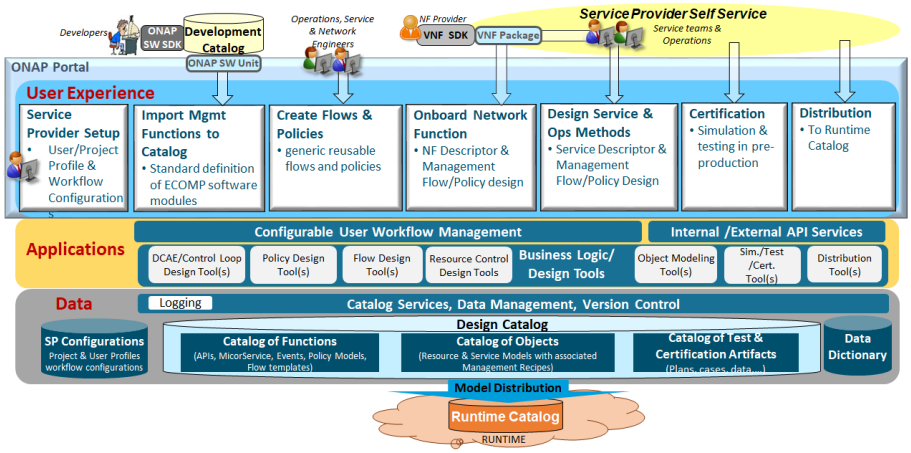


Figure 2.7: SDC Component [Open Network Automation Platform]

- Service Providers to configure its design environment including user roles and design workflows.
- Import generic ONAP management functions like, management system flows and policies from ONAP developed software and SP's adaptations.
- Onboard and Design resource level network functions (NFV, Physical Network Function)
- Add resources to service model composition.
- Design Service Provider-specific Management Flows and Policies for the Resource or Service Model.

- SDC unifies several design tools into a single platform, giving ONAP development a "Pluggable framework" for simple design tool integration.
- Provides design objects a common catalog with cataloging capabilities for storage and management of data models that follow the ONAP standard.
- Distributes models to runtime for execution.

Service Orchestration

The orchestration of network services and resources is the responsibility of the Service Orchestration (SO) component. Technology is based on the Business Process Model and Notation (BPMN) model. BPMN workflows can be separated into main processes and sub-processes. A BPMN workflow is a description, mainly composed of "tasks". Each task in a workflow is linked to a piece of code (Groovy or Java) that implements the task and they are executed by SO using the Camunda BPMN engine.

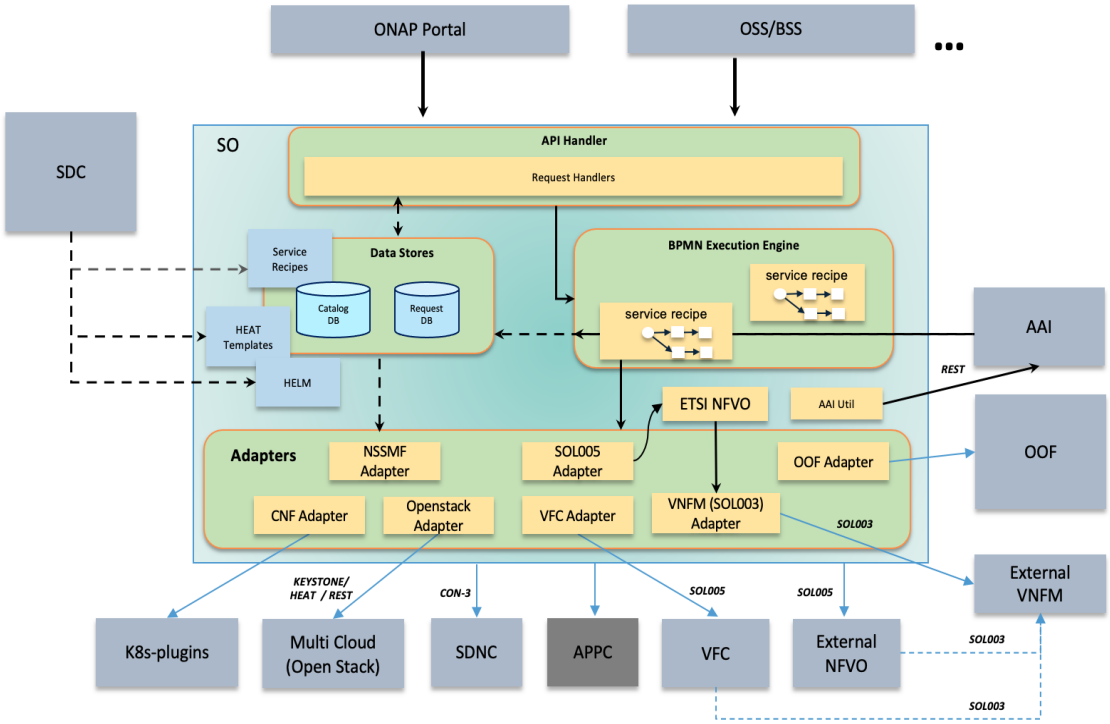


Figure 2.8: Service Orchestration Architecture [Open Network Automation Platform]

The architecture of this component can be viewed in figure 2.8 which consists of:

- **API Handler**

  Is a Representational State Transfer (RESTful) interface to northbound clients, which could be an external API, portal, etc., that handles service-level and infrastructure (NFV and network) requests. It is used to connect input requests to BPMN flows.

It is possible to keep track of open and finished requests by accessing Request BD allocated in Data Stores which will be explained further in the document.

- **BPMN Execution Engine**

  The main purpose of this component is to execute BPMN service recipes that are sent from the service orchestration. Sequence orchestration steps for each Resource in the recipe:

  - Request and configure network resources via Software-Defined Networking Controller (SDN-C);
  - Manage cloud resources via OpenStack;
  - Configure Application NFVs via Application Controller (APP-C);
  - Configure Network services via Virtual Function Controller (VFC);
  - Update inventory via Active and Available Inventory (AAI);

  BPMN Execution Engine also performs additional orchestration steps (consult policy, etc.) per individual recipes and error handling/rollback.

- **Resource Adapters**

  Resource adapters are components that are responsible for communicating with external systems and devices to provision and manage network resources.

  The Resource Adapters provide interfaces to lower-level controllers and other ONAP components, like:

  - Platform Orchestrator, SDN-C,APP-C, VFC, Multi-Cloud;
  - Hides the details of complex interfaces (e.g. OpenStack APIs) via higher-level calls;
  - Expose interfaces to BPMN flows as Simple Object Access Protocol Application Programming Interface (SOAP API) or Representational State Transfer Application Programming Interface (rest API);

  SO Catalog is used to map resource requests to a recipe/template. Catalog templates may be updated via self-service (outside of release cycles) and can be merged with input parameters at run-time.

- **Data Stores**

  Several different types of data stores can be used within ONAP, including relational databases, Not Only SQL (NoSQL) databases, and file-based storage systems. The specific data store that is used will depend on the needs of the particular ONAP deployment and the requirements of the various ONAP components that need to access the data. Data Stores are constituted by:

  - Request Database (DB): Tracks open and completed requests.
  - SO Catalog: enables service providers to more easily manage and deploy services

    * SO view of the SDC Catalog: Service and resource models, recipes, and templates;
    * Populated via SDC distribution service from TOSCA models;
  – Camunda DB: Maintains state for BPMN flows and can support multiple active engines.

- **SDC Distribution Client**

  The Service Design and Creation (SDC) component, which enables service providers to design and develop new network services using a visual designer tool, is often used in conjunction with the SDC Distribution Client.

  The SDC component creates a service artifact after a service provider completes developing a new service that includes all relevant data. The distribution of this artifact to the proper target environment, such as a development or production environment, is subsequently the responsibility of the SDC Distribution Client.

  SDC distribution client also receives distributions as Topology and Orchestration Specification for Cloud Applications (TOSCA) models, populates SO Catalog, and supports self-service updates to models and artifacts.

Active and Available Inventory

AAI has a centralized view of inventory data which includes changes from orchestrators, controllers, and assurance systems. The following things are under the control of AAI:

- Data Management: Giving the ONAP components access to the system's data, including its integrity, chronology, and sources, in addition to the current "as-built" view of the services. AAI includes information about the resources that are being used by the service provider, including details about the type, location, and status of each resource. This information is typically used to track the usage of resources, to monitor the performance of services, and to identify potential issues or problems.
- Inventory and Topology Management: As updates are made in the cloud, data in AAI is constantly updated in real-time.

Because AAI is metadata-driven, new resources and services can be readily added with the help of the Service Design and Creation (SDC) catalog definitions and the AAI model loader. A representation of AAI architecture is represented in figure 2.9

- AAI is updated with the details of new network or data center resources using REST APIs as they become available.
- Systems inform AAI at every stage of the deployment of new service types when they are created or new services are launched.

- The health and analytics findings are recorded in AAI as state information as telemetry for the services and their supporting infrastructure is gathered.
- The impact of a failure is evaluated using AAI when assurance systems identify it.

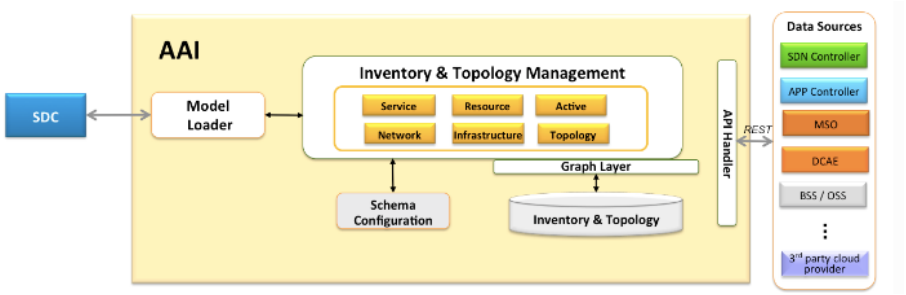


Figure 2.9: AAI[Open Network Automation Platform]

AAI must be ready to respond to a wide range of queries given the volume and diversity of the data, including:

- Real-time search, to quickly find specific items in all the information stored;
- Relationships to determine impacts and consequences;
- Aggregations and counts to investigate availability and consumption;
- Validation and Integrity to determine whether systems are acting on reliable information;
- History and provenance to reconstruct the current view and its context, and enrichment out to legacy systems.

Data Collection, Analytics, and Events

Data Collection, Analytics, and Events (DCAE) facilitate automation by collecting network data, performing analytics and correlation, and triggering events that can be used to identify root causes.

All of the microservices (collectors, analytics, and event processors) that provide active data flow and processing are included in the DCAE services components. These service components include functional entities that fulfill:

- A variety of data collection requirements;
- Event processors for data standardization;
- Analytics that evaluate acquired data;
- Supporting auxiliary microservices for automated closed loop flows.

2.5.2 MOSAIC5G

The established MOSAIC5G (M5G) PROJECT GROUP seeks to convert RAN and CN into open, flexible platforms for network-service delivery. With the help of such a platform, new use cases that would be interesting to various vertical businesses might be explored.

MOSAIC5G relies on 3 layers:

- Trirematics (Orchestration and Management);
- FlexRIC (RAN Platform);
- OpenAirInterface Radio Access Network (OAI-RAN) and OpenAirInterface Core Network (OAI-CN) (Infraestructure)

"On top of OpenAirInterface RAN and CN, this PROJECT GROUP will create a set of extensible control and orchestration frameworks with extensible APIs to allow for a fine-grained network infrastructure monitoring and programmability" [MOSAIC5G].

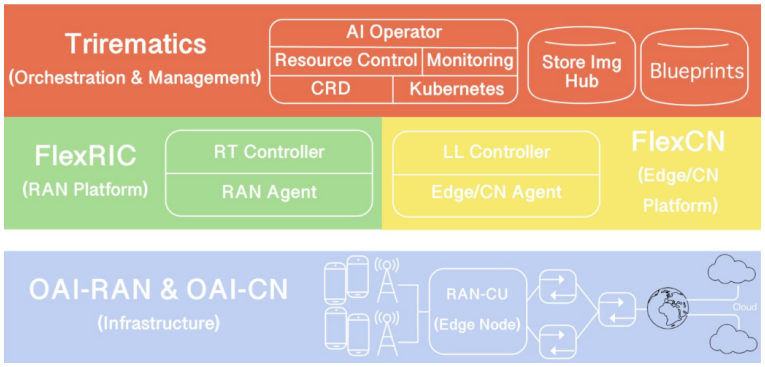


Figure 2.10: MOSAIC5G Architecture [MOSAIC5G]

In figure 2.10 we can have a better view of the MOSAIC5G architecture.

Since orchestration is the main emphasis of our topic, let's talk more specifically about trirematics. It is a cloud-native orchestration and management framework allowing to operate the lifecycle of diverse RAN and CN deployment scenarios in the form of blueprints.

In triremetrics, each network slice is independently optimized using distinct MOSAIC configurations for its resources, network functions, and service chains. Using several namespaces, these network slices are kept distinct from one another. Using REST northbound APIs, each slice may be managed and monitored similarly to ONAP.

The features of trirematics include:

- **Intelligence and Agility**

In instances requiring decision-making during deployment, the Trirematics operator demonstrates its intelligence by offering solutions for a range of tasks. These tasks encompass day-1 operations such as placement, resource allocation, and deployment, as well as day-2 operations which include activities like migration, fault recovery, parameter reconfiguration, and ongoing optimization.

- **Automation and Abstraction**

  The lifecycle of the network entities is fully automated by the Triremetric operator.

- **Maintenance and Observability**

  A 4G/5G network is anticipated to operate dependably for extended periods of time once it has been installed. With a wide range of observability features, such as log processing, alerting, metric processing, and health monitoring, the Trirematics operator could carry out automatic day-0, day-1, and day-2 operations.

AI Operator

Operators are software components that make operating another piece of software less complicated. They oversee a Kubernetes environment and use its present state to make decisions in real time, acting as an extension of the software vendor's technical team. Specifically, an operator automates the deployment of the services, upgrades, and (re)configures them.

This operator is developed using Openshift Operator Software Development Kit (SDK), which offers the resources needed to create, test, and evaluate them. SDK makes it easier to combine the business logic of an application (how to scale, upgrade, or backup), with the Kubernetes API to carry out such activities.

Different AI methods could be used for resource optimization and monitoring. After gathering and analyzing data from the deployed network, a decision on the provision or preemption of resources to/from particular slices can be made.

Kubernetes

Kubernetes is an open-source system to deploy, scale, and manage containerized applications anywhere.

As shown in figure 2.11, organizations used to run their programs on physical servers back in the traditional deployment period. With this method, the organizations are unable to define the amount of resources certain apps need to fulfil their requirement leading to problems with resource allocation. For instance, if multiple apps are running on a physical server, there may be times when one application uses up the majority of the resources, which lowers the performance of the other applications. A solution for this would be to run each application on a different physical server which was very expensive to organizations and was unable to scale if there were resources still available.

As a solution to this problem, virtualized deployment was introduced, allowing to run Multiple Virtual Machines on a single physical server's CPU. Because the

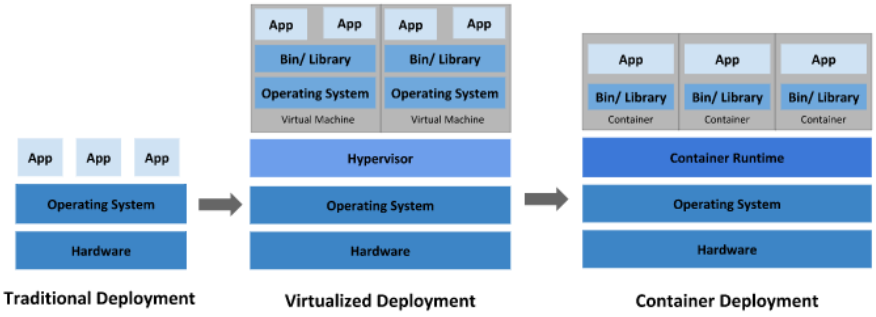


Figure 2.11: Deployment Evolution [KUBERNETES]

information of one application cannot be freely accessible by another application, virtualization enables applications to be segregated between VMs and offers a level of security. This method allows a more effective use of resources lowering hardware costs and making it easier to add or update applications.

With the exception of being able to access data between apps, Kubernetes' function is to manage containers, which is similar to virtual machines. Kubernetes provides a framework to run distributed systems. The applications deployed on a container are orchestrated by Kubernetes which takes care of the scaling and failover of these applications, provides deployment patterns, etc.

In the context of MOSAIC5G, the use of containers begins with the deployment of this project in a container orchestrated by Kubernetes which then we can provide auto-configuration, reconfiguration, and fault-tolerance.

The deployment of MOSAIC5G on Kubernetes is demonstrated in figure 2.12 using the M5G operator to allow the deployment of all MOSAIC 5G inside its own pods, Custom Resource Definition (CRD) which will define the composition of an Operator and an Operator life cycle that manages the installation, updates, and other stages of an operator's life cycle.

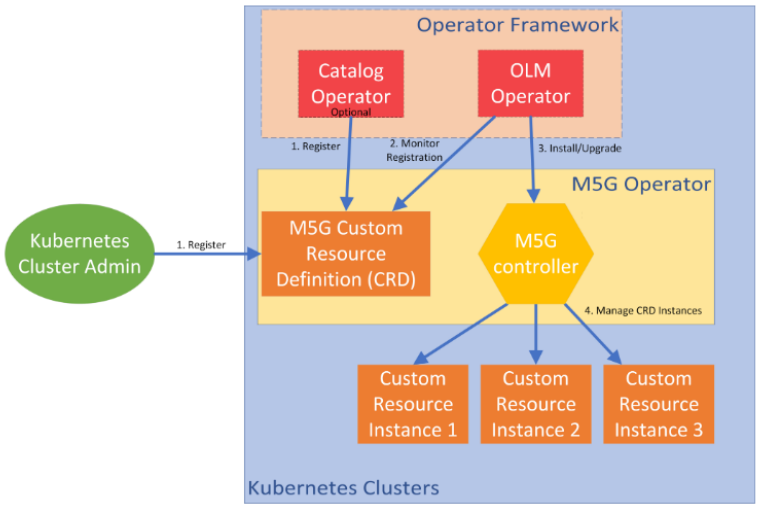


Figure 2.12: MOSAIC5G Architecture using M5G operator [Mosaic 5G on Kubernetes with M5G operator]

Custom Resource Definition

CRD provides the ability to add special objects or types to the Kubernetes cluster in order to satisfy their specific requirements. A Kubernetes CRD behaves exactly like any other Kubernetes object would: it makes use of all the ecosystem's features, its command-line interface, security, API services, and role-based access control.

CRDs are defined as blocks of data, that have the ability to offer a way to build, store, and expose Kubernetes API objects with data that meets any requirements that aren't automatically satisfied.

With the help of Mosaic5G Operator's auto-pilot function and custom resource definition CRD Kubernetes, we will be able to manage your services dynamically, including service upgrades and downgrades, network (re)configuration, etc.

Store Image Hub and Blueprints

These components are an NFV element manager, which is responsible for managing one NFV as a service while containing all the necessary interfaces to control the NFV's full lifetime and its connections to other NFVs and services.

The role of a blueprint, in particular, is to provide a detailed explanation of the organization, settings, and processes involved in setting up and maintaining a Network Slice Instance over its lifetime. A Network Slice Blueprint enables the design of a Network Slice that delivers particular network properties (e.g., ultra-low latency, ultra-reliability, value-added services for enterprises).

2.5.3 ONAP vs MOSAIC5G

Upon conducting a thorough study to understand both ONAP and MOSAIC5G platforms, we undertook a detailed comparison between them to explore alternative options for our implementation. In light of our analysis, we sought to find an orchestrator that could effectively handle network slicing without the excessive computational resource requirements of ONAP, leading us to explore MOSAIC5G as a potential solution.

The primary driver behind this exploration was the substantial difference in computational resource demands between ONAP and MOSAIC5G. ONAP, being a comprehensive platform for handling numerous network functions and services, necessitates a significant amount of computing resources, including CPU, memory, and storage, to ensure efficient and real-time data processing.

On the other hand, MOSAIC5G offers a lighter-weight approach, requiring fewer computational resources for orchestrating network slicing. This reduced resource footprint makes MOSAIC5G an attractive candidate, especially considering its potential to efficiently manage network-slicing tasks without overburdening the system.

While MOSAIC5G may have its own set of limitations and complexities based on the size and intricacy of the network being managed, it presents a viable and

promising option for our specific project requirements.

By exploring alternative orchestrators like MOSAIC5G, we aim to strike the right balance between efficiency and resource utilization, ultimately enabling us to achieve a robust and scalable network-slicing solution tailored to our needs.

2.6 Concept of fairness in resource allocation in network slices

The popularity of wireless technology has revolutionized our world by integrating almost everything into the Internet. This transformation can be observed through the beginning of technologies such as the Internet of Things (IoT), which involve the cooperation of a massive number of intelligent devices to provide a wide range of intelligent services. Ensuring fairness among these devices in this era of connectivity has emerged as a critical topic that must be examined from a variety of angles, including energy consumption, QoS, spectrum sharing, and more.

Due to their widespread use and the limited supply of resources, wireless networks in particular are a focus when we address the topic of fairness. With the exponential rise of wireless technology, the number of networked devices has expanded substantially, increasing the issues associated with resource allocation, especially when resources such as the wireless spectrum are few. Consequently, the issue of fair resource allocation and sharing becomes even more significant.

The problem of fair resource distribution in wireless networks is the focus of this section. We explore fairness research in the context of wireless networks, drawing ideas from the study [Fairness in wireless networks: Issues, measures, and challenges]. The paper offers a comprehensive look into fairness research, highlighting the subtleties involved in defining fairness and posing important queries about fairness measurement and system design.

2.6.1 Definition of fairness

Fairness in wireless networks is typically related to resource allocation or sharing. It ensures that resources are distributed equitably while considering the various needs and expectations of system users.

Unfair resource distribution can prevent some users or nodes from receiving essential resources (Resource starvation), which can have a negative impact on performance or the efficient use of resources. Fairness research has attempted to address this problem by creating plans and procedures that support equitable resource distribution. Depending on the needs and network architecture, these tactics can be applied to both centralized and distributed systems. With centralized methods, decisions about how to allocate resources are made by a central body using predetermined standards or formulas. Distributed systems, on the

other hand, provide each node the ability to bargain and work together to decide how to divide resources, providing fairness through group decision-making.

Establishing a fair resource allocation mechanism is the goal of fairness solutions. This mechanism should consider the various needs, preferences, and priorities of system users. Developing algorithms and protocols that consider elements like user requirements, resource availability, system limits, and network conditions is frequently required for this. Fairness solutions aim to balance increasing individual user pleasure with improving system performance as a whole.

2.6.2 Classification of Fairness Definition

Equal resource distribution among people is not a need for fair resource sharing or allocation. The concept of fairness can be categorized under some categories like equal opportunity, targeted fairness, resulting fairness, temporal fairness, and fairness for each task individually.

- **Equal opportunity:** Fairness in terms of equal opportunity focuses on making sure that everyone has an equal opportunity to acquire and use resources, however, does not ensure a fair distribution of resources. In a wireless network, for example, every node may have an equal chance to access a shared channel, but the actual channel time allocation may change depending on variables like network congestion or user demand.
- **Targeted fairness:** Targeted fairness, distributes resources in a way that fulfills particular demands or goals. For instance, in a network with several services, targeted fairness can demand giving priority to the allocation of resources to essential services or guaranteeing that each service satisfies a set minimum standard for quality of service.
- **Resulting fairness:** Resulting fairness refers to the fairness that was accomplished as a result of the resource distribution procedure. Instead of emphasizing opportunity equality, it concentrates on the final allocation outcome. To provide an equitable distribution of resources across nodes in the wireless network scenario, the resource allocation might be dynamically adjusted based on the current network conditions.
- **Temporal Fairness:** It is acknowledged by temporal fairness that resource distribution in dynamic systems can change over time. It considers fairness in various temporal situations and recognizes the dynamic nature of resource availability. For instance, in a wireless network, temporal fairness may entail assigning resources based on changing user needs throughout the day or periodically redistributing resources to respond to changing traffic patterns.
- **Fairness per service:** Fairness can be assessed separately for each task in a network where nodes carry out multiple functions or offer diverse services. For instance, in a wireless ad-hoc network (self-configuring network of mobile devices (nodes) that communicate without a fixed infrastructure), the

allocation of bandwidth for various services can be used to measure fairness and guarantee that each service gets a fair share of resources.

2.6.3 How to measure fairness

Fairness measures play a vital role in evaluating and quantifying the level of equity in resource allocation or sharing scenarios. They provide objective and numerical representations of fairness, enabling comparisons between different allocation schemes. In this section, we explore both quantitative and qualitative fairness measures and their significance in guiding decision-making processes.

Quantitative Fairness Measures

Quantitative fairness measures are metrics used to evaluate and quantify the level of fairness in resource allocation or sharing scenarios. These measures provide numerical representations of fairness, allowing for objective and quantitative comparisons between different allocation schemes. They play a crucial role in assessing the fairness of resource distributions and guiding decision-making processes. These metrics need to meet a number of criteria, such as continuity, individual independence, and scalability. Jain's index [A quantitative measure of fairness and discrimination] and Shannon's entropy [The mathematical theory of communication] are some examples of quantitative fairness measures. Jain's Index is a ratio that illustrates how fairly resources are distributed, with a higher value signifying a fairer distribution, and Shannon's entropy might show fairness and quantify the uncertainty of resource distribution when the entropy value is larger.

Fairness can also be assessed using other measures of variability, such as the ratio or difference between the greatest and lowest values of a performance indicator. Less frequently used in fairness studies are variables like unfairness and variability metrics like the Lorenz curve [Methods of measuring the concentration of wealth] and the Gini coefficient [Variabilita e mutabilita].

Qualitative Fairness Measures

Without offering a numerical representation, qualitative fairness metrics concentrate on evaluating fairness. Max-Min Fairness is frequently used in qualitative metrics.

Max-Min Fairness

Max-Min fairness attempts to distribute as much as possible to customers who pay cheap rates, with minimal resource waste. Single-path rate allocation is a straightforward illustration of max-min fairness, where a network is made up of links with defined capacities, source-destination pairs communicate with one another over a single predetermined path, each source-destination pair must be given a rate and the rate on each link is kept below its capacity.

A rate allocation is considered fair in the sense of max-min fairness if it is impossible to raise the rate of one flow without also lowering the rate of a lower flow.

Finding the max-min fair vector in resource allocation issues is done using the Max-Min Programming (MP) Algorithm [Radunovic and Le Boudec, 2007]. The algorithm produces the max-min fair allocation by repeatedly maximizing the smallest coordinate of the feasible vector until all coordinates are fixed.

- **Max-Min Programming (MP) Algorithm**

    1. **Identify the smallest component:** The algorithm begins by examining the max-min fair vector and identifying the smallest component within it.
    2. **Maximize the minimal coordinate:** Maximize the minimal coordinate by adjusting the corresponding allocation to its highest possible value within the constraints of the system, while keeping the other allocations unchanged.
    3. **Fix the minimal coordinate:** Once the minimal coordinate is determined, it is fixed at its maximum value. This means that the allocation corresponding to the minimal coordinate is set to its maximum allowable value, ensuring that no other allocation can be increased beyond this point.
    4. **Remove the corresponding dimension:** After fixing the minimal coordinate, the algorithm removes the dimension corresponding to that coordinate. This means that the focus of the algorithm shifts to the remaining coordinates, excluding the one that has been fixed.
    5. **Repeat steps 1 and 2:** The algorithm continues by repeating steps 1 and 2 on the reduced set of coordinates. It identifies the new smallest component among the remaining coordinates and fixes it at its maximum value.

2.7 Summary

In this chapter, we discussed several technologies that facilitate the implementation of network slicing, including MEC, NFV, Cloud Computing, and SDN. These technologies play a crucial role in enabling network slicing, and we provided a brief description of how they work and their contributions to the implementation of network slicing.

The chapter also covered the fundamental graph analysis and optimization tools that form the basis of our resource allocation framework for network slicing. These tools, such as NetworkX for network analysis, MiniZinc for mathematical modeling of resource allocation, and Matplotlib for visualizing outcomes, empower efficient resource allocation, fair network slicing, and valuable insights into network structures and performance.

Furthermore, a comparison between 5G and 4G networks was performed to understand the advancements and capabilities brought by 5G technology. The analysis focused on improvements in speed, latency, capacity, and energy efficiency. The architecture of 5G networks and their various components, including the role of elements like NG-RAN, were explained in delivering high-performance network services.

In a crucial aspect of our research, network slicing was addressed comprehensively. The objective was to provide a clear overview of the concept and its practical implementation. We began by defining network slicing, explaining its key characteristics, and highlighting its significance in the context of next-generation networks. Real-world applications of network slicing across various services were presented to showcase its versatility and potential impact on various domains.

The chapter concluded with a comparison between ONAP and MOSAIC5G, focusing on their ability to individually monitor network slices and manage their life cycle. The analysis explored the components and functions of each platform, highlighting key differences and similarities in their monitoring and management capabilities. The aim was to provide insights into selecting an appropriate platform for network slice monitoring and life cycle management based on specific requirements and constraints.

Additionally, this chapter discussed the concept of fairness in wireless networks, with a focus on resource allocation and sharing. With the rapid growth of wireless technology, ensuring fairness among intelligent devices in the IoT era has become critical. The section emphasized the urgent problem of fair resource distribution in wireless networks and explored fairness research, drawing insights from a comprehensive review of fairness in wireless networks.

Fairness in wireless networks was defined as the equitable distribution of resources while considering the diverse needs and expectations of system users. The section categorized fairness definitions into five types, including equal opportunity, targeted fairness, resulting fairness, temporal fairness, and fairness per service. It highlighted that unfair resource distribution can lead to resource starvation and negatively impact performance and resource utilization. Fairness solutions aim to create plans and procedures to support equitable resource distribution, either through centralized decision-making or group decision-making in distributed systems.

Overall, the chapter provided valuable insights into the implementation of network slicing, the advancements brought by 5G technology, real-world applications of network slicing, and a comparison of platforms for network slice monitoring and life cycle management. It also sheds light on the importance of fairness in wireless networks, the various definitions of fairness, and the measures used to assess fairness in resource allocation and sharing scenarios.

Chapter 3

Related Work

3.1 Works for 5G and Use Cases

In this section, we present a selection of real projects and use cases that demonstrate the practical applications of 5G technology and network slicing. These projects serve as valuable references for our research, as we leverage them to formulate test cases and validate the capabilities of our framework. By studying these real-world implementations, we gain insights into how 5G and network slicing can address challenges and enhance network functionalities across diverse domains, including smart city infrastructure, critical communications, disaster management, healthcare, and more. These projects provide a tangible link to our work, guiding us in developing robust and relevant test scenarios to assess the efficacy of our framework in practical settings.

3.1.1 OREOS - Orchestration and Resource Optimization for Reliable and Low-latency Services

This section presents a compelling use case involving the OREOS project, which aligns closely with the activities proposed in this dissertation. The insights presented in this section are based on the valuable research of [OREOS: Orchestration and Resource optimization for rEliable and lOw-latency Services] and [OREOS: Estudos Preliminares na área do Projeto].

The OREOS project stands as an exemplary case study that provides valuable insights into the practical implementation of 5G network slicing. This project's comprehensive research offers a deep understanding of the diverse applications of network slicing in the context of fifth-generation (5G) mobile communications.

The primary objective of the OREOS project is to develop and deploy an end-to-end orchestration platform capable of providing and managing critical services in 5G networks. These services encompass a wide range of applications, including vehicle communications, electrical distribution networks, and emergency communications for public and private entities.

A key focus of the OREOS project is to address cutting-edge technological challenges related to 5G network development, particularly in supporting services based on URLLC. This capability is crucial, as it paves the way for revolutionary changes in sectors like automotive and smart cities.

The OREOS project encompasses two major use cases to validate its platform: the Smart City use case and the Autonomous Driving use case. Here, we focus on the Smart City use case, which includes three user stories: pedestrian safety, air quality management, and crime prevention.

The Smart City use case is set in an urban environment, aiming to enhance people's safety, improve the quality of life through environmental monitoring, and increase the efficiency of crime prevention actions.

In the first user story, pedestrian safety is a critical concern, particularly at road crossings. To address this, video cameras throughout the city are installed to detect pedestrians at the crossings or their intention to cross. This may be accomplished through intelligent mechanisms that are able to detect people, pedestrian crossings, and their movement without having to alert approaching vehicles to slow down or stop before reaching pedestrians. One example of this mechanism is, that if vehicles are connected to the city network, the use of network information can notify drivers in advance of a pedestrian's presence on the roadways. This scenario is represented in figure 3.1

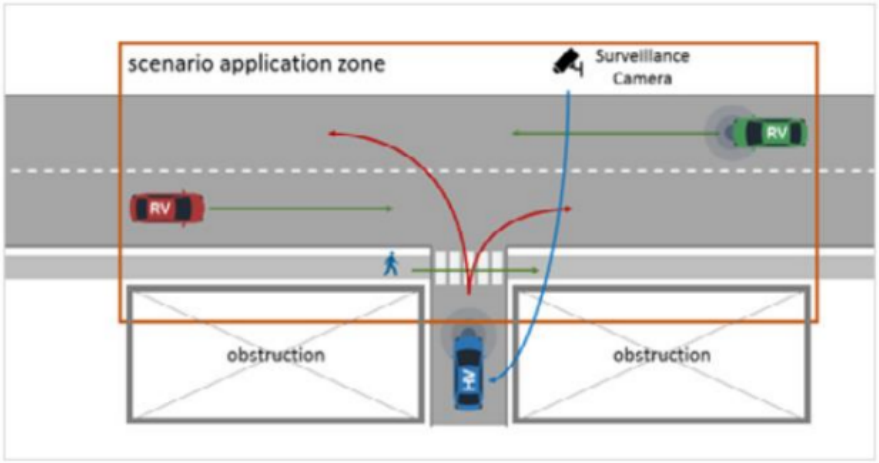


Figure 3.1: Pedestrian Crossing [de Coimbra, 2021b]

The second user story involves air quality management, which requires a setup with multiple vendors connecting sensors and devices to gather data on weather and emission rates. The collected data is analyzed in quasi-real-time and sent to various cloud platforms for predictive weather forecasting and real-time response to pollution incidents. Citizens are provided with this information to raise awareness and potentially avoid highly polluted areas.

The third user story revolves around crime prediction, enabling agile policing, and enhancing visitor safety. The use of machine learning models identifies malicious actions, such as robberies, based on police force reports, generating risk

information for different neighborhoods. Citizens and visitors can access this information through applications, helping them make informed decisions about their safety.

The Smart City use case's objectives are tailored to each user story:

- **Pedestrian Safety:**
  - Enable pedestrian detection at road crossings.
  - Support real-time video analysis.
  - Facilitate real-time information sharing with vehicles.
  - Provide feedback to pedestrians about approaching vehicles.
- **Air Quality Monitoring:**
  - Collect data from multiple sensors and IoT devices.
  - Aggregate data for edge analysis.
  - Provide feedback on air quality measurements and predictions.
  - Integrate pollution information with navigation systems.
- **Crime Prevention:**
  - Detect robberies through video analysis or other sources.
  - Predict crime based on historical data.
  - Inform users about risks through city-specific applications.
  - Collect data from citizens and tourists about their perception of safety.

The feasibility of this use case is strongly influenced by the potential of 5G network slicing by distributing the different user stories across distinct slices, optimizing their performance and efficiency.

Figure 3.2is a good example of how different slices are represented in the context of the OREOS project. As we can see, each slice has its own unique qualities in accordance with its purpose.

With the technologies discussed in the previous chapter, the concept of Ultra-Reliable Low Latency Communication (URLLC) slice comes into focus, emphasizing the critical criteria of low latency and high reliability. To effectively meet these criteria, a strategic deployment approach becomes essential, and this is where Mobile Edge Computing (MEC) servers play a pivotal role.

By strategically placing MEC servers in key locations, especially in highly busy streets frequently used by pedestrians, the network gains significant advantages. The incorporation of MEC servers at these strategic spots enhances the network's cloud computing capabilities, thereby fulfilling the high-demanding requirements of 5G. The result is a substantial reduction in latency for critical services, ensuring swift and reliable communication.

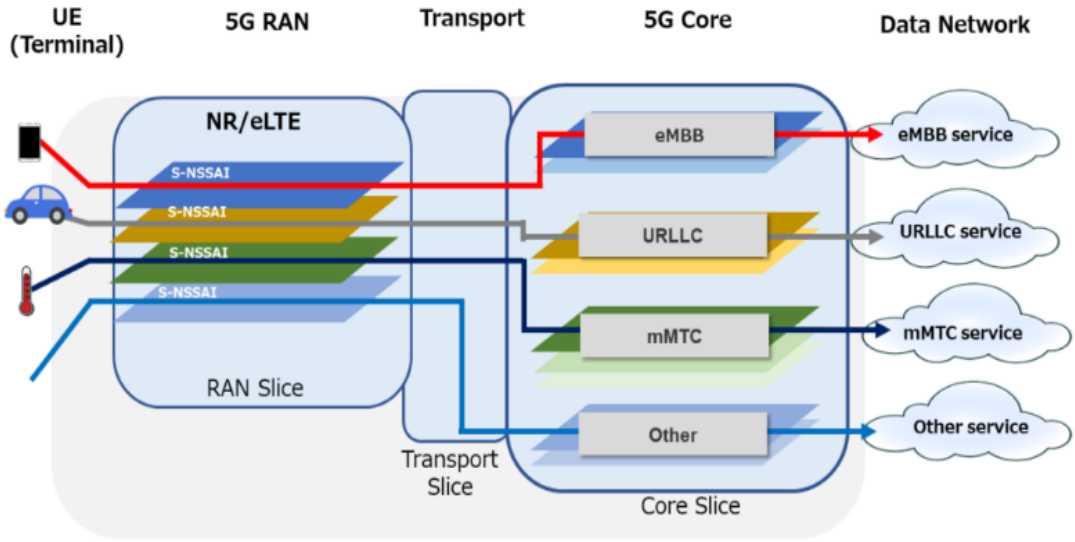


Figure 3.2: End to End Network Slicing Overview [What is 5G? Differences between 4G and 5G (eMBB, URLLC, mMTC)]

With the URLLC Slice powered by MEC infrastructure, smart cities can achieve a seamless and secure environment for pedestrian safety. Video cameras equipped with intelligent mechanisms can detect pedestrians crossing streets and their movements. By leveraging the MEC-enabled network, information about approaching vehicles can be instantly shared, allowing timely alerts to pedestrians and drivers to avoid potential collisions. This real-time communication between pedestrians and vehicles contributes to enhanced safety at crossings.

Furthermore, the MEC infrastructure empowers smart cities with efficient air quality monitoring. By deploying a Massive Internet of Things (MIoT) Slice, sensors and IoT devices can gather and aggregate air quality data in real time. This data is then analyzed at the edge, providing accurate air pollution information and enabling predictive weather forecasting. The seamless integration of pollution data with navigation systems empowers citizens to make informed decisions to avoid routes with high pollution levels.

Figure 3.3 illustrates how a MEC server might influence QoE:

Lastly, the Enhanced Mobile Broadband (eMBB) Slice ensures robust crime prevention measures. The eMBB infrastructure supports high-capacity and high-speed communication, which is essential for agile policing and ensuring the safety of city dwellers. Machine learning models can efficiently detect and report malicious actions within neighborhoods, providing valuable insights into crime patterns and enhancing overall security.

Taking this information into account, we represent each user story with a specific Service and Slice Type (SST) characteristic, as follows:

- **Pedestrian Safety:** URLLC (Ultra-Reliable Low Latency Communication) slice.

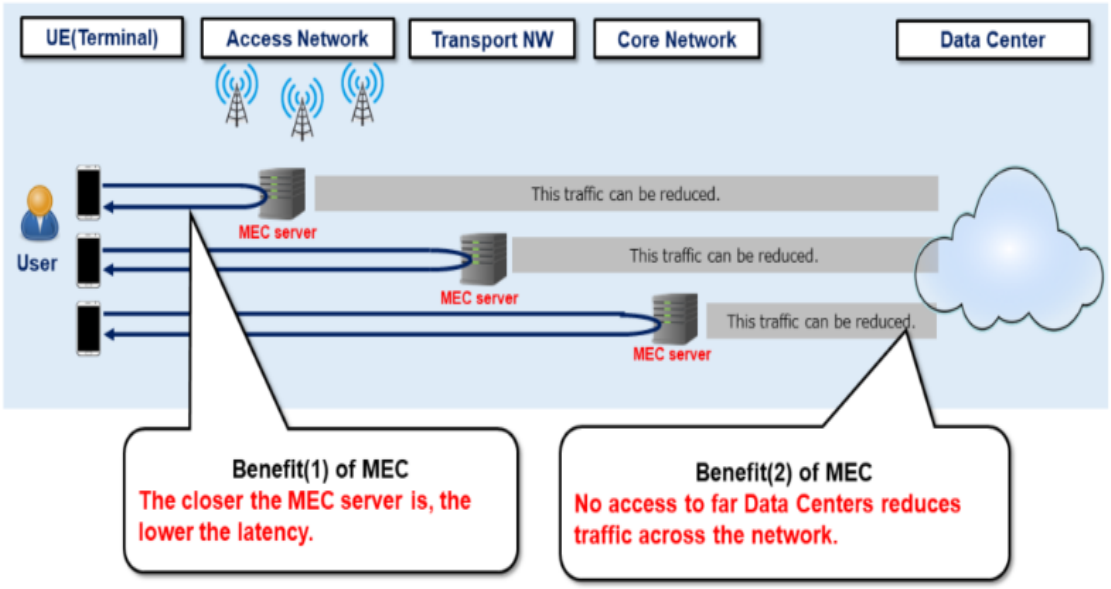


Figure 3.3: Physical Installation Cases for MEC servers [What is 5G? Differences between 4G and 5G (eMBB, URLLC, mMTC)]

- **Air Quality Monitoring:** MIoT (Massive Internet of Things) slice.
- **Crime Prevention:** eMBB (Enhanced Mobile Broadband) slice.

In alignment with our research, we will leverage these specific use cases to structure our test case regarding this project.

Testing Environment

The testing environment for the Smart City use case is a simulated and controlled setup replicating a real urban city context. It focuses on validating the functionality, performance, and security of the smart city infrastructure. The components used in the topology include:

- **Cloud Clusters:** Various cloud clusters represent different components of the smart city infrastructure, such as AAI, AAF, SDN-C, SO, DCAE, Policy, and SDN-R. These clusters handle management, orchestration, and policy enforcement.
- **5G Core Cluster:** This critical component includes AUSF, NSSF, PCF, AMF, NRF, and SMF, responsible for core network functions like authentication, policy control, and network slicing to ensure seamless communication and services for 5G-enabled devices.
- **OpenStack:** The test environment utilizes OpenStack, an open-source cloud computing platform, to manage and control the cloud clusters, providing virtualization capabilities and efficient resource management.
- **User Equipment (UEs):** Virtual nodes or simulated devices represent UEs, which simulate end-user devices in a real-world 5G network context. These devices connect to the network to access services and applications.

- **Edge Cluster:** The Edge cluster includes components like bNB, UPF, N3WF, FlexRIC, FlexCN, and AF. It enables low-latency services and enhances network performance at the network edge.
- **Smart City Applications:** The test environment deploys city-specific applications responsible for pedestrian safety, air quality monitoring, and crime prediction. These applications run as virtual nodes or servers, providing the respective functionalities.

The logical abstraction of the topology involves using 25 nodes to represent the entire architecture, with each node representing a specific component. It is essential to note that this representation does not imply the physical deployment of individual nodes; in a real-world scenario, some components may run on the same physical servers or virtual machines based on resource allocation and deployment strategies. Figure A.1 illustrates the topology diagram, providing a visual representation of the test case setup.

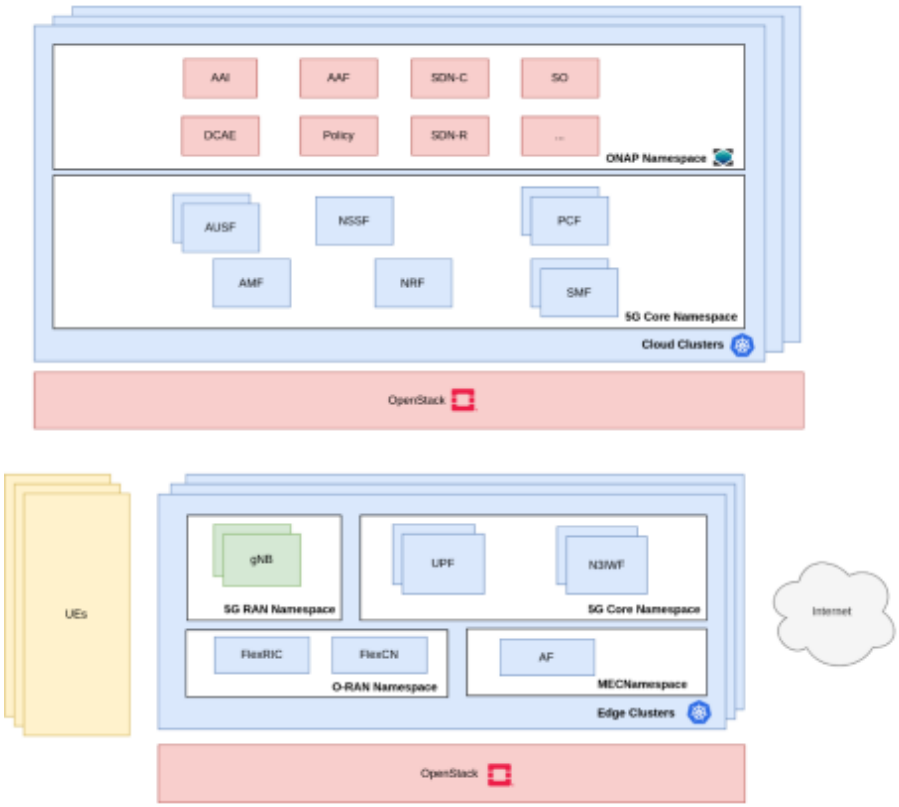


Figure 3.4: OREOS topology[OREOS: Metodologia de Avaliação e Especificação de Casos de Uso para Validação]

3.1.2 Network Slicing for Critical Communications in 5G Communication

In recent years, the emergence of modern technologies, such as Smart Grids driven by renewable energy resources, Intelligent Transportation Systems (ITSs) with

self-driving cars, and the IoT, has led to increased demands on Information and Communication Technology (ICT). Critical Infrastructures (CIs) play a crucial role in supporting these advancements, necessitating dedicated communication networks to meet their specific and divergent requirements.

However, the traditional approach of deploying separate communication infrastructures for each critical use case incurs high costs and lacks scalability. In response to these challenges, the concept of 5G network slicing has been introduced, and its definitions and advantages were already explained in the section 2.4.

To fulfill the performance targets of 5G and support the diverse requirements of CIs, this project's approach to network slicing builds on the principles of NFV and SDN, which are instrumental in addressing the dynamic demands of Critical Infrastructures, providing the ability to reconfigure communication networks according to specific service requirements.

In the network slicing architecture suggested in this research, authors use queuing methods, in particular Hierarchical Token Bucket, in combination with SDN and NFV. The Open vSwitch serves as an open-source virtual multilayer switch deployed on DP devices in the network. Our SDN-MANO controller acts as an orchestrator, dynamically creating and managing individual slice controllers tailored to specific use cases. Each slice is represented as a separate bridge containing virtual ports nested within the main bridge.

When traffic enters the data plane, the MANO controller classifies packets based on their protocol or other criteria supported by the OpenFlow protocol. Packets are then assigned to the appropriate slice bridge, and the slice's controller routes the packets to their virtual destination port. This routing process is repeated at each hop to the destination, ensuring traffic flows are mapped to the appropriate QoS queue and physical port in the main bridge. Best-effort traffic is classified for flows that do not match any slice, ensuring efficient resource utilization.

This project's approach is designed to be adaptable to different technologies beyond Ethernet, supporting both wired and wireless (air interface) 5G communication. This flexibility ensures compatibility with a wide range of Critical Infrastructure applications, addressing diverse data rates, latency, reliability, and other requirements.

We employed this project as a means to validate our fairness model, utilizing real-world data to imbue it with practical significance. The testing environment, delineated as follows, serves as the backdrop for this validation process:

Testing Environment

The test setup consists of **13 identical servers**, each equipped with a four-core, 2.2GHz Intel Xeon D-1518 CPU, 16GB of RAM, and six 1GBaseT Ethernet ports featuring different Network Interface Cards (NICs). The servers operate on Ubuntu Server 16.04.3 LTS as their operating system.

Scenario Components

- 3 Virtual switches form a sliced data plane responsible for efficiently forwarding and switching network traffic within the shared infrastructure.
- 3 designated servers act as hosts and are responsible for generating and processing network traffic over the sliced network.
- 4 SDN controllers participate in the scenario. 3 controllers utilize Floodlight (v1.2) as their software platform and are assigned to manage specific slices: Smart Grid, Intelligent Transport, and Multimedia Services.

  The fourth controller runs Ryu (v4.19) as its software platform and serves as an SDN-MANO controller, coordinating and managing the slices established by the Floodlight controllers.

The fig 3.5 is a visual representation of this scenario.

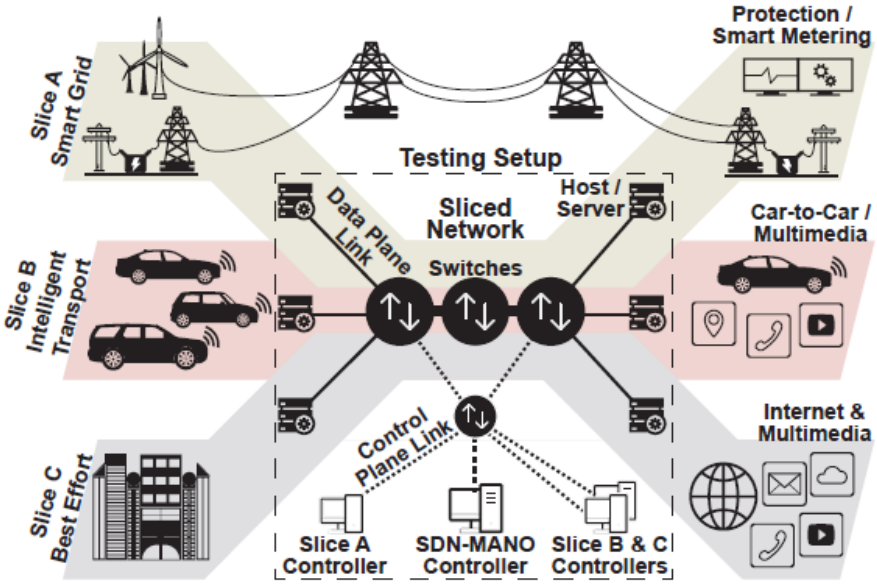


Figure 3.5: Evaluation Scenario within the Testing Setup [Network slicing for critical communications in shared 5G infrastructures-an empirical evaluation]

3.1.3 Service Function Chaining in Wildfire Scenarios

Wildfires in Mediterranean countries, such as Portugal, pose significant risks during summer and demand efficient fire combat strategies. Mission Critical Services (MCS) play a vital role in coordinating first responders' human and technical resources to combat wildfires effectively. However, traditional communication technologies like Public Mobile Radio systems fall short of meeting MCS's stringent Quality of Service (QoS) requirements. As a solution, the authors of this article, propose an innovative approach that harnesses the power of Service Function Chaining and Fog Computing to enhance MCS capabilities and ensure reliable and timely communication during critical wildfire situations.

The dynamic and challenging nature of wildfire combat scenarios requires a modernized approach to communication. Traditional Public Mobile Radio technologies lack the necessary capabilities to fulfill MCS's diverse requirements, hindering real-time information exchange and decision-making. This research seeks to address these limitations by leveraging emerging technologies, such as 5G, LTE, and Fog Computing, to optimize MCS communication infrastructure and services.

The Conceptual Framework of the proposed project outlines the fundamental principles and components that drive the enhancement of MCS in wildfire combat scenarios. It emphasizes the significance of Service Function Chaining in orchestrating different Service Functions to achieve optimal performance. These Service Functions are classified based on their relevance to MCS and include Mission Critical Voice, Mission Critical IoT, multimedia services, location data, and Mission Analysis. The primary objective is to improve situational awareness by integrating real-time video streaming and sensor data from the field.

The integration of Fog Computing and SDN is a crucial aspect of the project. Fog nodes deployed at edge, intermediate, and cloud levels play a pivotal role in collecting sensor data and managing devices. Fog Computing enables efficient data processing at the edge of the network, reducing the need to send all data to centralized cloud servers. SDN dynamically configures network paths to optimize MCS resilience and ensure timely communication during wildfire situations. By combining Fog Computing and SDN, the project aims to create a flexible and scalable communication infrastructure that can adapt to changing firefighting requirements.

The development of effective Service Function Chaining policies is essential to streamline data flow and minimize latency. Different approaches for mapping SFs onto Virtual Machines at the edge and cloud nodes are introduced in this section. These policies are designed to maximize operation time and reduce processing overhead, taking into account the computational capabilities of each node. By efficiently chaining SFs, the project aims to enhance the overall performance of MCS, ensuring that critical services are delivered with low latency and high reliability.

This article played a pivotal role in shaping our approach to validating the fairness model. Specifically, we leveraged the insights within this article to construct and simulate a series of comprehensive test cases. By doing so, we were able to procure real-world values pertaining to crucial service characteristics, such as bandwidth, latency, CPU utilization, and memory requirements. These values, extracted from the context provided by the article, not only substantiated the practical application of our fairness model but also enriched its accuracy and effectiveness in real-world scenarios.

3.1.4 5G Smart City Lighting

The MATILDA and SliceNet projects jointly define a 5G Smart City Lighting network slicing architecture by implementing an end-to-end operational service

framework that covers the life cycle of design, development, and orchestration of 5G network services through a One-Stop API over a programmable infrastructure. All of this information regarding the following section was found in [Rusti et al., 2019b] and [Rusti et al., 2019a].

Applications based on microservices can be independently orchestrated across this programmable infrastructure with the aid of metadata from application graph components, which give indications about the infrastructure requirements.

The way that these projects work is that information regarding the provided requirements is mapped to the slice intent MATILDA framework, which results in the creation of the appropriate slice intent description.

This descriptor is then sent to the One-Stop API made available by SliceNet, to fulfill the request by responding with an application-aware slice object. An end-to-end slice instance is established once the management plane has received and processed the slice intent sent through the SliceNet One-Stop API.

5G Smart City Lighting is automated controlled and efficiently managed, with real-time and historical energy consumption measurements, as well as real-time detection of failures, energy loss, or energy theft potential.

3.2 Models for fairness

3.2.1 SALEM: Service Fairness in Wireless Mesh Environments

Wireless Mesh Networks (WMNs) have undergone rapid evolution, enabling seamless connectivity for a wide array of intelligent devices and small-sized sensors, regardless of their physical location or time of day. The ubiquitous support of wireless networks has introduced features that enhance the quality of services, such as high availability rates, efficient energy usage models, and intelligent spectrum management. However, the growing adoption of WMNs has accentuated the need for effective resource management and fairness in resource allocation to ensure equal access to network resources for all devices.

In response to these challenges, SALEM is proposed, a novel approach to address the fairness routing problem in WMNs. The main objective of SALEM is to distribute network resources fairly, considering three key fairness objectives: reliability, delay, and energy consumption.

To achieve this, SALEM employs a Mixed Integer Linear Programming (MILP) formulation, which mathematically models the fairness objectives as optimization problems. The formulation involves defining binary variables to represent the allocation of services on links and calculates the normalized values for reliability, delay, and energy fairness. These objectives may be conflicting, as different paths may offer varying levels of reliability, delay, and energy consumption. To address this challenge and obtain a compromise solution, SALEM utilizes a min-max formulation that assumes all objectives are equally relevant.

The SALEM heuristic, a central component of the approach, is implemented in the ONOS SDN controller. It leverages the JGraphT library [JGraphT] to model the network topology as graphs and uses the *CapacityScalingMinimumCostFlow* algorithm [Applications of network optimization] to find optimal paths for each service. The algorithm calculates the normalized weight for each link based on the fairness objectives and ensures that services traverse paths that provide sufficient flow capacity with minimal costs. The obtained paths are appended to a list, which contains all the routes for the requested services.

The implementation of SALEM in the ONOS SDN controller enhances the practicality and real-world deployability of the proposed solution. Its capabilities to manage routing decisions and improve fairness in resource allocation make it a promising approach for future WMNs. The SALEM project contributes valuable insights to the field of WMNs, providing a fairness-driven solution for routing optimization that aligns with the evolving demands of wireless communication technologies.

We will leverage the SALEM approach from this article to test our fairness model in real-case scenarios. The primary objective of this test scenario is to evaluate the performance of our fairness model in a wireless mesh network (WMN) within a real smart-city environment.

In this scenario, we aim to emulate a complex WMN topology, encompassing diverse wireless technologies such as LTE, 5G, mmWave, WiFi, and Gigabit Ethernet. This topology draws inspiration from the Aveiro Tech City Living Lab (ATCLL) situated in Aveiro, Portugal, known for its cutting-edge technological infrastructure.

Testing Environment

The testing environment for the network slicing scenario is emulated using the Mininet SDN network emulator. Mininet allows us to create a virtual network environment that closely resembles a real network, enabling comprehensive testing and validation of the network-slicing architecture. The environment is structured based on the following components:

- **2 centralized backbone SDN switches (swcc)**: Represent the central controllers for the network slices in a multi-slice environment. They act as the orchestrators for resource allocation and service provisioning across the entire infrastructure. Each switch is responsible for managing a specific slice, ensuring that the resources are efficiently distributed among the services within that slice.
- **6 clusters switches**: Connect the edge switches within a particular slice, forming the internal communication structure of that slice. Each cluster is connected to a dedicated SDN switch (swc).
- **36 edge switches**: Each Cluster switch, contains 6 edge switches (sw). the edge switches are responsible for connecting the end devices (such as Raspberry Pi 3B or Cubiboard 3 machines) to the backbone switches. They handle the traffic flow within each slice and ensure that services within the same

slice can communicate efficiently.

Raspberry Pi 3B and Cubiboard 3 Devices represent the nodes that are part of the network slices. Raspberry Pi 3B devices are used as central and cluster switches, while Cubiboard 3 devices are employed as edge switches.

The logical abstraction of the topology involves using 44 nodes to represent the entire architecture, with each node representing a specific component. Figure 3.6 illustrates the topology diagram, providing a visual representation of the test case setup.

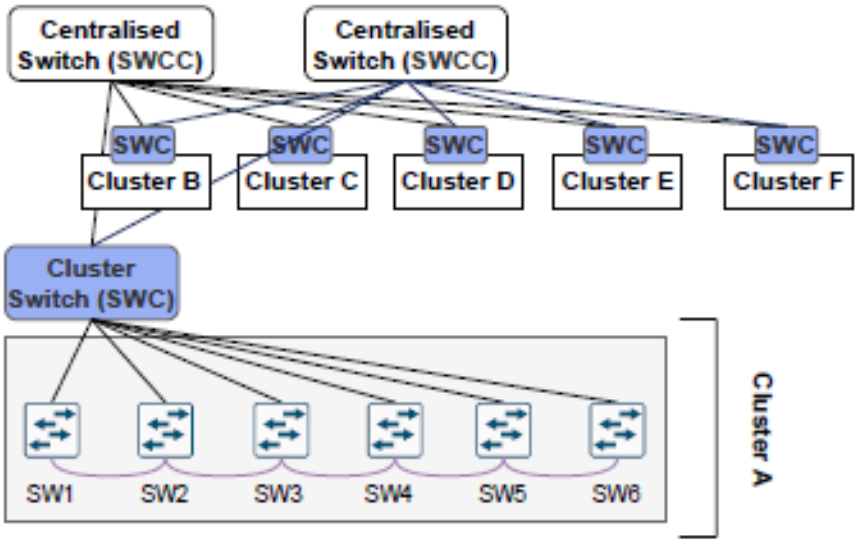


Figure 3.6: Topology diagram [SALEM: Service Fairness in Wireless Mesh Environments]

3.3 Summary

In the "Related Work" chapter, we explore a selection of projects and use cases that highlight the practical applications of 5G technology and network slicing across various domains. These real-world implementations serve as essential references for our research, guiding us in the formulation of test cases and the validation of our fairness model. By closely studying these projects, we gain valuable insights into how 5G and network slicing can effectively address challenges and improve network functionalities in areas such as critical communications, smart city infrastructure, etc.

The insights obtained from these projects were instrumental in constructing real-world test scenarios to assess the performance of our fairness model. By adapting the network topologies and service requirements described in these articles, we were able to create relevant and robust test cases that reflect the practical complexities and demands of modern communication networks. By using the scenarios presented in these works, we designed our test scenarios adapted to our

fairness model, ensuring that our evaluations align with the intricacies of real-world communication environments.

Chapter 4

Research Objectives and Approach

This chapter outlines the key goals of this study, as well as the approach employed and the decisions made. The research methodology is presented in section 4.1, and the research objectives are identified and described in section 4.2.

4.1 Research Methodology

This dissertation began with a study of what a 5G network is, how it works, and what customers may acquire from it. An analysis was done to explore these principles and some of the technology utilized to operate this network. This analysis was conducted with the help of digital libraries like IEEE Explore, where a number of papers and articles were filtered based on the research emphasis.

5G network softwarization and slicing, 5G network slicing enabling technologies and orchestrators, were some keywords searching for the concepts in the articles mentioned above. The publications used in this research, as previously noted, were filtered by the year 2018 and above using digital libraries.

Additionally, all the information regarding orchestrators' architecture was gathered from their official websites, particularly ONAP and MOSAIC5G. This information made possible a better view of the components used in each orchestrator.

Implementing network slicing in an orchestrator requires significant time and resources, which are often limited. As a result, we adjusted our strategy to focus on resource allocation and sought a solution to address this challenge. Our selected solution was NetworkX, a tool that enabled us to create a simulated network closely resembling real-world conditions. This simulation provided us with the opportunity to apply our fairness model within the context of network slicing.

To effectively utilize NetworkX and generate a realistic network topology, we delved into the intricacies of its operation. This exploration was guided by an array of scholarly works, including references such as [5G Dataset Network Slicing CRAWDAD Shared], [Nerini, 2020], and [Resilient service chains through smart replication]. Through these sources, we gained profound insights into the tool's

functionalities and its vital role within the realm of network analysis and simulation.

After the investigation and understanding of the concepts and technologies behind the 5G Network, a study focused on the architecture of this network was made using solutional videos [MPIRICAL 5G Core Network] which provided a thorough analysis of the various components of the architecture of this Network.

4.2 Research Objectives

The objectives of this thesis include:

- **Identification of current network slicing approaches and requirements.**

  As part of this study, we examined current network slicing implementations in order to identify the various methodologies employed. This provided us with a deeper understanding of how different approaches, which all operate within the same concept, can function in various contexts. Due to the complexity of the topic, particularly in regard to understanding the application of network slicing in smart cities, this analysis was crucial in gaining a comprehensive understanding of its exploration.

- **Validation of an optimization model for network slicing.**

  This research seeks to validate the effectiveness of a fairness model designed for network slicing. This model plays a crucial role in optimizing how network resources are allocated and how network slices are configured to meet the unique needs of various users and services. Prioritizing fairness means that every network slice gets a fair and suitable portion of resources. Validating this model is significant because it ensures that it actually works well, and is practical, effective, and reliable in real-world situations.

- **Test and evaluation of the optimization model for network slicing**

  The aim of this objective is to conduct a comprehensive examination and assessment of the fairness model that has been validated, specifically in the context of achieving a fair resource allocation within network slicing. This evaluation involves rigorous testing, utilizing authentic data derived from real-world scenarios, thereby confirming the model's validity.

  The outcomes obtained from these meticulous tests served a twofold purpose: firstly, to measure the actual effectiveness of the fairness model and its ability to deliver on its promises; and secondly, to ensure its practicality and usability within real-life situations. This objective carries substantial significance, as it establishes the model's robustness and utility, verifying its potential applicability to tangible projects that require equitable resource distribution.

  It's important to highlight that the testing and evaluation process involved using data gathered from a wide range of use cases and the development of

complex simulations to replicate real-world scenarios. This multifaceted approach guaranteed a thorough evaluation that accurately assesses the model's performance in practical, real-world contexts.

- **Implementation of network slicing in an orchestration platform.** As previously discussed, successful network slicing relies on a technology that can meticulously oversee each distinct slice. This level of oversight is achieved through orchestrators who possess an intricate understanding of the network's dynamic elements. Our objective was to put the validated optimization model into practical use within an active orchestrator. This practical implementation aimed to provide firsthand insight into how the model performs in a real-world context. However, it's important to note that due to the substantial computational resources needed for the implementation of network slicing within an orchestration platform, **this specific objective has not been executed**.

4.2.1 Approach

This section outlines the approach taken to achieve the objectives outlined in Section 4.2. The approach for achieving the designed goals is elaborated upon in this section.

- **Identification of current network slicing approaches and requirements.**

  To achieve this objective, a review of multiple publications related to the theme was conducted, considering the perspectives of different authors in order to gain a comprehensive understanding of the current state of network slicing and the various approaches that have been proposed and implemented.

  Case studies, such as the OREOS project, were also analyzed to understand the approaches being used in real-world environments and their performance. In addition, practical experimentation was conducted to gain firsthand experience with the various network slicing approaches and assess their strengths and limitations. Together, these approaches provided us with a comprehensive understanding of the current state of network slicing and the various approaches being used.

- **Validation of an optimization model for network slicing.**

  This thesis aims to investigate the optimization of network slicing by validating a fairness model that maximizes the fairness of bandwidth and CPU for each slice. The goal of this research is to make better use of the resources available in a network and improve the overall performance of network slicing.

  This model was validated in the MiniZinc platform, a constraint modeling language, where was tested under different scenarios to evaluate its performance. The results of the simulations were used to validate the effectiveness of the model and to identify any potential improvements.

- **Test and evaluation of optimization models for network slicing.**

  Following the validation process within the MiniZinc platform, the model was transitioned to real-world contexts for implementation. These scenarios have been crafted using the NetworkX library, enabling us to construct network topologies that closely resemble actual networks. By simulating real-world network conditions, we rigorously tested and evaluated our model, ensuring its competence in managing authentic data. This validation process substantiates its applicability in real projects that require a resource allocation model, thus affirming its practicality in real-world scenarios.

- **Implementation of network slicing in an orchestration platform.**

  This objective remained unfulfilled due to the substantial computational resources and time investment needed to develop a fully functional orchestration platform. Despite the challenges that prevented us from realizing this goal directly, we proactively sought alternative approaches to replicate the intended scenario. Our solution involved leveraging the capabilities of the NetworkX library, enabling us to simulate a network environment that closely emulates a real-world setting. While a comprehensive orchestration platform couldn't be realized, this innovative approach allowed us to explore and study network slicing dynamics within a simulated environment, thereby contributing to our understanding of the concept's practical application.

4.2.2 Plan

In this chapter, we present the planning for the first semester in Figure 4.1 and the second semester in Figure 4.2 of the project. In this Gantt charts, we can visualize the distribution of tasks to ensure that the project is completed within the given timeframe.

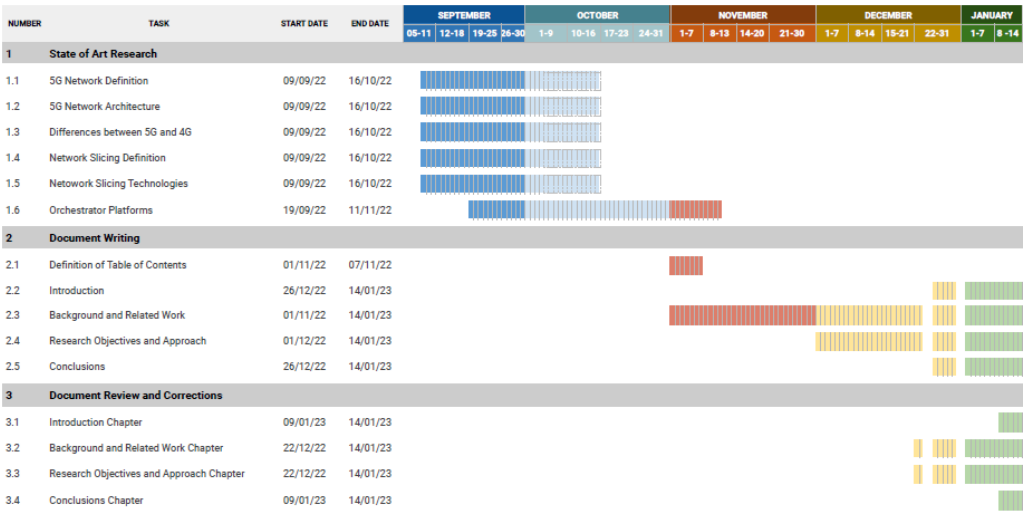
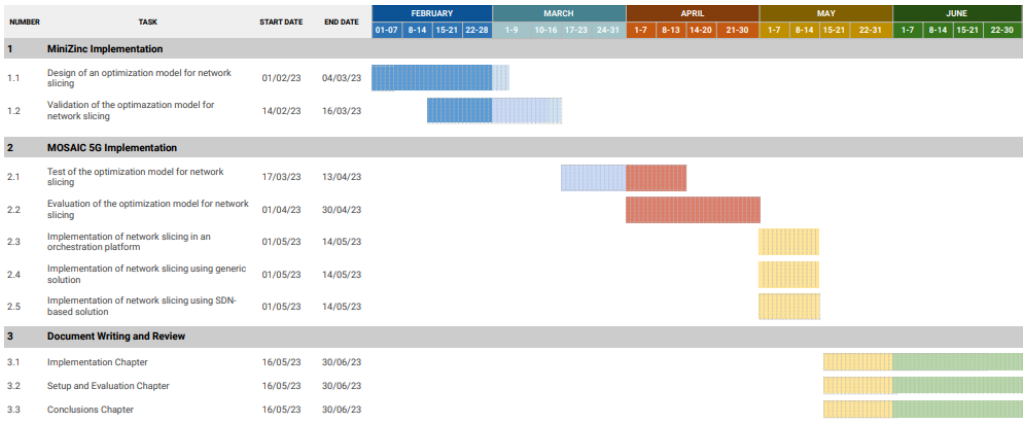


Figure 4.1: First Semester Plan



NUMBER	TASK	START DATE	END DATE
1	**MiniZinc Implementation**		
1.1	Design of an optimization model for network slicing	01/02/23	04/03/23
1.2	Validation of the optimazation model for network slicing	14/02/23	16/03/23
2	**MOSAIC 5G Implementation**		
2.1	Test of the optimization model for network slicing	17/03/23	13/04/23
2.2	Evaluation of the optimization model for network slicing	01/04/23	30/04/23
2.3	Implementation of network slicing in an orchestration platform	01/05/23	14/05/23
2.4	Implementation of network slicing using generic solution	01/05/23	14/05/23
2.5	Implementation of network slicing using SDN-based solution	01/05/23	14/05/23
3	**Document Writing and Review**		
3.1	Implementation Chapter	16/05/23	30/06/23
3.2	Setup and Evaluation Chapter	16/05/23	30/06/23
3.3	Conclusions Chapter	16/05/23	30/06/23

Figure 4.2: Second Semester Plan

4.2.3 Risks

This section aims to identify and assess the potential risks that may be encountered in the pursuit of achieving the objectives outlined in Section 4.2. By identifying and evaluating these risks, we can develop strategies for reducing or addressing them in order to ensure the success of the project.

One of the notable risks identified during the course of this study pertains to the inherent complexity of the ONAP orchestrator platform. As highlighted earlier, the deployment and validation of network slicing strategies within the ONAP environment demand substantial computational and time resources. In order to address and mitigate this risk, we adopted an alternative strategy. Specifically, we opted to simulate a realistic network topology using the versatile capabilities offered by the NetworkX library. This simulation approach allowed us to replicate essential aspects of a real network environment, enabling us to explore network slicing dynamics and evaluate our model's performance without the constraints posed by the ONAP platform's resource demands.

4.3 Summary

In this chapter, we have outlined the key objectives and approach taken in our research. We began by delving into the complexities of the 5G network, its principles, and technologies through an extensive review of academic literature and digital libraries such as IEEE Explore. Additionally, we gathered insights into orchestrator architectures, particularly from ONAP and MOSAIC5G official sources, to better understand their components.

Our initial plan involved implementing network slicing within an orchestrator. However, due to resource limitations, we adjusted our approach to focus on resource allocation, utilizing the NetworkX tool to create realistic network simulations closely resembling real-world conditions. This shift allowed us to apply and validate our fairness model within network slicing more efficiently.

The research objectives were identified and defined in detail in this chapter:

- **Identification of current network slicing approaches and requirements:** We conducted a comprehensive review of existing network slicing implementations and methodologies, including real-world case studies such as the OREOS project.
- **Validation of an optimization model for network slicing:** We developed a fairness model and validated it within the MiniZinc platform under various scenarios to ensure its effectiveness.
- **Test and evaluation of the optimization model for network slicing:** We transitioned our model to real-world scenarios using the NetworkX library, conducting extensive testing and evaluation to confirm its practicality and reliability.
- **Implementation of network slicing in an orchestration platform:** While this specific objective was not executed due to resource constraints, we employed NetworkX to simulate network slicing within a realistic environment, contributing to our understanding of its application.

We also provided a visual representation of the project plan for the first and second semesters, as well as an assessment of potential risks, with a particular focus on the complexity of the ONAP orchestrator platform.

In conclusion, this chapter has set the stage for our research by outlining our goals, approach, and objectives. We have adapted our methodology to address resource limitations and have identified potential risks. This groundwork will guide our research throughout the project, ensuring its successful completion.

Chapter 5

Fairness Model in the Resource Allocation Framework

In this chapter, we introduce the Fairness Model, a crucial component of our project focused on optimizing resource allocation in network slicing. To develop the Fairness Model, we drew upon the concepts and insights discussed in Chapter 2, where we explored the notion of fairness, its measurements, and the various types of fairness, such as max-min and min-max, both qualitatively and quantitatively.

Chapter 3 presented a collection of relevant research articles that played a pivotal role in shaping our approach. We utilized the values and parameters presented in these articles to construct realistic test cases for validating and fine-tuning the Fairness Model.

Our primary objective in building the Fairness Model was to achieve equitable resource allocation among different network slices. By prioritizing fairness, we sought to ensure that each slice receives a balanced and appropriate share of bandwidth and CPU resources.

To achieve this, an objective function was formulated, aiming to maximize fairness while considering the requested bandwidth and CPU for each service. The model's constraints were thoughtfully designed, reflecting real-world constraints and requirements observed in the related research.

As we mentioned earlier the structure and formulation of this fairness model were already established, so our objective is to validate the viability of this model for integration into other projects that require a resource allocation algorithm. One of these projects is the OREOS project which was already introduced in chapter 3, where we're focusing on validating that the model can indeed be used effectively.

Throughout this chapter, we will delve into the structure and components of the Fairness Model, showcasing its integration of insights from Chapter 2 and the utilization of data from Chapter 3. By combining these resources, we have created a robust and reliable model that optimizes resource allocation in network slicing while ensuring fairness across services. The comprehensive approach taken in

this chapter allows us to demonstrate the model's efficiency and its potential for practical implementation in 5G networks.

5.1 Objective Function

Optimization is a common feature in mathematical programming models. By maximizing or reducing an objective function, these models identify the ideal response to a particular problem. The quantity that we wish to maximize or minimize is represented by the objective function. In our model, the objective function aims to maximize the fairness of bandwidth and CPU for each slice. Instead of focusing on individual flows, the aim of this objective is to consider the overall used resources in each slice in a fair manner.

Our model's objective function considers two crucial variables: CPU and bandwidth allocation. It considers the requested bandwidth and CPU for each service, assigning weights and penalties to facilitate allocation decisions.

To evaluate the overall objective function 5.1, we sum up the bandwidth and CPU contributions of all services and slices. The solution aims to minimize this value, thereby reducing the overall cost. The model reaches for optimal performance by balancing the utilization of bandwidth and CPU, ensuring adherence to slice and service constraints.

$$\min \max_{s=1,\ldots,S} FR(x,y) = \sum_{a=1}^{A} \left(\sum_{i,j=1}^{n} \left(1 + W(1 - q^{as}\bar{q}_{ij}^{s})\right) x_{ij}^{as} b^{a} + \sum_{i=1}^{n} \left(1 + W(1 - q^{as}\bar{q}_{i}^{s})\right) y_{i}^{as} C^{a} \right) \tag{5.1}$$

To understand better the objective function of our model we can divide it into two parts, Bandwidth and CPU contribution.

- **Bandwidth Contribution:** This part of the equation, calculates the bandwidth contribution to the objective function for each service "a" within slice "S":

$$\sum_{i,j=1}^{n} \left(1 + W(1 - q^{as}\bar{q}_{ij}^{s})\right) x_{ij}^{as} b^{a} \tag{5.2}$$

  $(1 + W(1 - q^{as}\bar{q}_{ij}^{s})$, is a weight factor that penalizes the allocation of services on links that are not in the slice q_{ij}^{s} or not part of the service q^{as}.

  x_{ij}^{as} represents whether service "a" is allocated on the link (i, j) and is multiplied by the requested bandwidth (b^{a}) for service "a". The entire expression is summed over all pairs of nodes and weighted by the penalization factor.

- **CPU Contribution:** This part of the equation, calculates the CPU contribution to the objective function for each service "a" within slice "S":

$$\sum_{i=1}^{n} (1 + W(1 - q^{as}\bar{q}_i^s))\, y_i^{as} C^a \tag{5.3}$$

$(1 + W(1 - q^{as}\bar{q}_i^s)$, is a weight factor that penalizes the allocation of services on nodes that are not in the slice $\bar{q}_i^s$ or not part of the service q^{as}.

y_i^{as} represents whether service "a" is allocated on node i, and is multiplied by the requested CPU (C^a) for service "a". The entire expression is summed over all pairs of nodes and weighted by the penalization factor.

Table 5.1 represents all the parameters and the variables used in the objective function, in order to understand the impact of each one in the fairness result.

Parameters	**Description**
S	Number of slices
A	Number of services
n	Number of nodes
b^a	Requested bandwidth by service a
C^a	Requested CPU by service a
W	Weight parameter
q^{as}	1 if service a is assigned to slice s and 0 otherwise
q_{ij}^s	1 if slice s uses link (i, j) and 0 otherwise
$\bar{q}_i^s$	1 if slice s uses node i and 0 otherwise
Variables	**Description**
x_{ij}^{as}	Binary variable that is 1 if link (i, j) is used by service a and 0 otherwise.
$y_i^a s$	Binary variable that is 1 if node i has service a allocated and 0 otherwise.

Table 5.1: Variables and Parameters in Fairness Model

In some instances, it may be relevant to share resources between slices. For example, each service belongs to a specific slice but that same slice may be overloaded and is not able to allocate resources. Thus, we introduce a weight parameter W to apply that penalization, which changes the objective function value for a given node/link. The penalisation W will be applied if any of the previous parameters (q^{as}, q_{ij}^s and $\bar{q}_i^s$) is 0. If the penalization is not applied, the equation will be 0. This is solved by increasing 1 the result of the penalization.

5.1.1 Linearization

Linearization is a mathematical technique employed in this thesis to address the problem of optimizing fairness in bandwidth and CPU allocation for each slice.

The central objective is to achieve fair resource distribution among the various slices of the system.

The optimization problem, as represented by Equation 5.1, initially involves a nonlinear objective function. However, to make it suitable for efficient computational methods, the function is linearized. This transformation is accomplished by introducing an auxiliary variable, denoted as Z, which serves as a critical component in the linearization process.

In the linearized form, the objective function seeks to minimize the auxiliary variable Z. By doing so, the fairness of resource allocation is maximized, aiming to reduce any potential imbalances or discrepancies in the distribution of resources across the slices. This results in a more fair and efficient resource utilization strategy for the entire system.

The optimization problem is subject to the linear constraints as defined in the equation:

$$\begin{aligned} &\min Z \\ \text{s.t. } &Z \geq FR(x,y), \ \ \forall s = 1,\ldots,S \end{aligned} \tag{5.4}$$

Here, Z acts as a real variable that represents the fairness measure for resource allocation among the slices. The inequality constraints ensure that Z remains greater than or equal to the fairness measure ($FR(x,y)$) for all individual slices ($s = 1,\ldots,S$).

The objective function is to maximize the fairness of bandwidth and CPU allocation for each slice. The auxiliary variable Z is introduced in order to linearize this objective function. The original objective function is transformed into a minimization problem with linear constraints. The goal is to reduce the imbalance or discrepancy in resource distribution across the slices by reducing Z.

One of the significant advantages of employing linearization is the simplicity and interpretability it brings to the model. Linear functions are computationally efficient and easy to analyze, making them a suitable choice for optimizing resource fairness in this context.

5.1.2 Min-Max Fairness

We have adopted the min-max approach to achieve fairness in our model. This approach is designed to minimize the maximum allocation among the participating entities, ensuring a well-balanced distribution of resources. By preventing situations where certain services receive disproportionately larger resources than others, the min-max fairness approach fosters resource equilibrium, mitigates resource concentration, and addresses disparities. Consequently, this approach enhances the overall fairness and equity of resource allocation within the network-slicing environment.

The min-max fairness approach is implemented through the introduction of the real variable Z in the linearized version of the objective function. The optimiza-

tion problem seeks to minimize Z while ensuring that Z is greater than or equal to the fairness metric FR(x, y) for all slices (s = 1, ..., S).

Minimizing Z is all about maximizing fairness. It might seem a bit counterintuitive, as we're using the term "minimize," which typically suggests making something as low as possible. However, Z here represents a crucial measure of fairness.

In essence, Z stands for the minimum fairness metric among all the network slices under consideration. So, when we use the term "minimizing Z", we're striving to minimize the differences in fairness metrics across these slices.

To simplify this concept, our goal is to elevate the fairness level across all slices, even in the worst-case scenario, where fairness might be at its lowest. By minimizing Z, we're effectively closing the gap between the fairness metrics of different slices, striving for a higher and more uniform level of fairness.

In essence, the term "minimizing Z," refers to minimizing the discrepancies in fairness among slices, ultimately leading to a better, more balanced distribution of resources. This approach ensures that no single slice is unfairly favored, promoting a fair allocation of resources to all slices, even in the toughest cases.

While it's true that our focus on the worst-case scenario may not always align with real-world situations, the advantages offered by the min-max approach make it an appealing option for resource allocation in network slicing.

It is important to note that the design of the model itself was beyond the scope of our work, and thus, we solely focused on its implementation aspect. Fortunately, the min-max fairness metric was already an integral part of the existing design, aligning perfectly with our objectives.

The deliberate choice of the min-max fairness approach is attributed to its inherent simplicity and ease of practical application. Its underlying concept is straightforward, enabling seamless integration into the existing resource allocation model. The mathematical formulation and linearization can be effortlessly incorporated into the solver or relevant equations used for optimization. This streamlined integration process ensures a smooth and effective implementation, minimizing the complexity associated with configuring and deploying resource allocation algorithms.

Moreover, the computational efficiency of the min-max fairness approach is highly advantageous for real-world network slicing challenges. Other, more complex fairness measures may demand substantial computational overhead, potentially limiting their practicality, particularly in large-scale networks with multiple slices and services. In contrast, the simplicity and efficiency of the min-max fairness approach make it a viable and effective solution for resource allocation in such complex network environments. Its ability to strike a balance between fairness and computational efficiency empowers the model to cater to the diverse resource requirements of different slices while optimizing network performance and user experience.

5.2 Model Constraints

In the context of optimization and mathematical programming, constraints are conditions or limitations that restrict the feasible solutions of a problem. These constraints define the boundaries within which the optimization process must operate to find the most favorable solution that satisfies all the given requirements.

In the Fairness Model for resource allocation within network slicing, constraints play a pivotal role in shaping the optimization process. They impose specific limitations on the allocation of resources, ensuring that the resulting solution adheres to various predefined criteria. These criteria could include avoiding over-utilization of resources, meeting latency requirements for certain services, or maintaining flow conservation within the network.

In this section, we explore the constraints that govern the Fairness Model, aimed at optimizing resource allocation within network slicing. These constraints play a crucial role in ensuring fairness and efficiency in resource utilization. We will delve into resource constraints to prevent over-utilization, flow constraints for valid routing, and delay constraints for meeting latency requirements. Understanding and applying these constraints will lead to a more balanced and robust resource allocation strategy, meeting the diverse needs of different slices and services.

5.2.1 Resource constraints

The model's resource constraints are made to make sure that the overall bandwidth, CPU, and memory requests for each slice and node don't go above their allowed amounts. By preventing resource overuse, these restrictions are essential for ensuring fairness.

The restriction on the overall CPU allocation for each node is imposed by Equation 5.5:

$$\sum_{s=1}^{S} \sum_{a=1}^{A} y_i^{as} C^a \leq \bar{C}_i, \ \forall i = 1, \ldots, n \tag{5.5}$$

$\bar{C}_i$ corresponds to the total CPU in node i, which according to this constraint, the total amount of allocated CPU for all services in each slice must be lower than or equal to. This restriction promotes fairness in CPU consumption by guaranteeing that the CPU resources are distributed evenly and preventing any single slice or service from monopolizing the CPU capacity.

The total amount of RAM that each node receives is constrained by Equation 5.6:

$$\sum_{s=1}^{S} \sum_{a=1}^{A} y_i^{as} M^a \leq \bar{M}_i, \ \forall i = 1, \ldots, n \tag{5.6}$$

This constraint, makes sure that the overall amount of memory allocated for all services in each slice M^a does not exceed the node's entire available memory capacity $\bar{M}_i$. This restriction prevents resource overutilization and guarantees equitable memory consumption across various slices by limiting memory allocation.

For each link, Equation 5.7 limits the total requested bandwidth:

$$\sum_{s=1}^{S}\sum_{a=1}^{A} x_{ij}^{as} b^a \leq \bar{b}_{ij},\ \forall i,j = 1,\ldots,n \tag{5.7}$$

$\bar{b}_{ij}$ corresponds to the total bandwidth in link (i,j). This constraint defines that the total bandwidth allocated to all services in each slice must be less than or equal to the link's total bandwidth capacity, ensuring that the bandwidth resources are allocated equally among the slices and preventing any slice or service from using excessive amounts of bandwidth, which could affect the performance of other slices.

5.2.2 Flow constraints

Here it is assumed that a predefined source and target exist for each flow. Moreover, it is also necessary to ensure that each service will be allocated in a node existing in the path.

The flow conservation concept is maintained by Equation 5.8:

$$\sum_{s=1}^{S}\sum_{j=1}^{n} x_{ij}^{as} - \sum_{s=1}^{S}\sum_{j=1}^{n} x_{ji}^{as} = U_i^a,\ \forall i = 1,\ldots,n,\ a = 1,\ldots,A \tag{5.8}$$

This constraint states that the difference between the incoming and outgoing flows of each service in each node should be equal to a predetermined value U_i^a which indicates if node i is the source (1), transport (0) or target (-1) for a given service a. By satisfying this constraint, the model ensures that the allocated services follow a valid path from their source to the target, respecting the predefined routing requirements. It maintains the integrity of the flow by guaranteeing that the allocated services are properly routed within the network.

The allocation of services within each node is constrained by Equation 5.9:

$$y_i^{as} \leq \sum_{j=1}^{n}(x_{ij}^{as} + x_{ji}^{as}),\ \forall i = 1,\ldots,n,\ a = 1,\ldots,A,\ s = 1,\ldots,S \tag{5.9}$$

This constraint states that the number of services allocated in each node y_i^{as} should be less than or equal to the sum of the incoming x_{ij}^{as} and outgoing x_{ji}^{as} flows for

those services. By enforcing this constraint, the model ensures that services are allocated in nodes that exist in the path from the source to the target. It prevents the allocation of services in nodes that are not part of the valid path, which could lead to incorrect routing and unreliable service delivery.

To ensure that each link in the path is used in a single direction we implement Equation 5.10:

$$\begin{aligned} x_{ij}^{as} + x_{ji}^{as} \leq 1, \ \ \forall i, j = 1, \ldots, n, \\ a = 1, \ldots, A \\ s = 1, \ldots, S \end{aligned} \tag{5.10}$$

This constraint is used, to prevent the model from selecting disconnected links in the path. It ensures that the allocation of links for a specific service a in a slice S follows a unidirectional path and avoids using the same link for both forward and backward directions. If the sum of x_{ij}^{as} and x_{ji}^{as} is greater than 1 (i.e., 2 or more), it means that both directions of the link between nodes "i" and "j" are being used simultaneously. This violates the unidirectional constraint, as a service a should only be allowed to traverse the link in one direction, either from node i to node j (forward) or from node j to node i (backward). By enforcing this constraint, we ensure that for each service a in slice s the link between nodes i and j is used in only one direction.

5.2.3 Delay constraint

Due to the nature of some services, it is necessary to ensure a given end-to-end latency for the service. For example, in a URLLC slice, latency is of most importance.

$$\sum_{s=1}^{S} \sum_{i,j=1}^{n} x_{ij}^{as} d_{ij} \leq \bar{d}^{a}, \ \ \forall a = 1, \ldots, A \tag{5.11}$$

In order to enforce a maximum end-to-end latency for particular services, the model's delay constraint 5.9 is applied. It makes sure that these services don't have their latency requirements violated by excessive delays introduced by the resources and pathways that were provided to them.

Finally, we need a constraint to ensure that each service will be allocated only once.

$$\sum_{s=1}^{S} \sum_{i=1}^{n} y_{i}^{as} = 1, \ \ \forall a = 1, \ldots, A \tag{5.12}$$

$$
\begin{aligned}
y_i^{as} \leq q_i^s, \quad & \forall i = 1, \ldots, n, \\
& a = 1, \ldots, A \\
& s = 1, \ldots, S
\end{aligned} \tag{5.13}
$$

$$
\begin{aligned}
x_{ij}^{as} \leq q_{ij}^s, \quad & \forall i, j = 1, \ldots, n, \\
& a = 1, \ldots, A \\
& s = 1, \ldots, S
\end{aligned} \tag{5.14}
$$

5.3 Summary

In conclusion, the Fairness Model presented in this chapter plays a critical role in optimizing resource allocation within the network slicing context. The model is designed to prioritize fairness by ensuring that each slice receives an equitable share of resources, leading to a more balanced and efficient allocation strategy.

The objective function of the Fairness Model aims to maximize the fairness of bandwidth and CPU allocation for each slice. By considering the requested bandwidth and CPU for each service and introducing weights and penalties for allocation decisions, the model optimizes resource distribution while adhering to slice and service constraints.

To transform the initial nonlinear objective function into a computationally efficient form, the concept of linearization is applied. By introducing an auxiliary variable, the model achieves the objective of minimizing the fairness measure, thereby maximizing fairness in resource allocation across slices.

The Fairness Model is governed by a series of constraints to maintain equitable resource distribution. Resource constraints prevent over-utilization by limiting the total CPU and memory allocated for each node and the total bandwidth allocated for each link. Flow constraints ensure that services are allocated in nodes along valid paths, maintaining the flow conservation concept. Additionally, delay constraints guarantee that specific services adhere to their end-to-end latency requirements.

By effectively addressing these constraints, the Fairness Model optimizes resource allocation in a way that promotes fairness among different slices while meeting the diverse requirements of services.

In summary, the Fairness Model is a vital component of our project that strives to achieve equitable and efficient resource allocation within the network slicing framework. By prioritizing fairness in bandwidth and CPU allocation and adhering to various constraints, our model enhances the overall performance and reliability of the network while meeting the unique demands of different slices and services.

Chapter 6

Network Resource Allocation Framework

In the context of modern networking, efficiently allocating network resources poses a critical challenge. With the escalating demand for diverse services and applications, conventional static network configurations often fall short of delivering optimal performance. To address this, dynamic and flexible resource allocation techniques have emerged, aiming to optimize resource utilization while ensuring fair distribution among different services and users.

Numerous methods and models have been proposed to tackle the resource allocation problem in networks. Each approach comes with its own advantages and limitations, underscoring the importance of identifying the most suitable technique for a specific network scenario. Existing approaches may prioritize specific performance metrics, such as maximizing throughput, minimizing latency, or reducing energy consumption. While these metrics are crucial in their respective contexts, they might not offer the comprehensive perspective required to address all resource allocation challenges effectively.

This section introduces a comprehensive framework meticulously designed to optimize network resource allocation while upholding fairness as a key criterion. Unlike many existing methods that focus on individual flows or specific performance metrics, our framework adopts a comprehensive approach, considering the overall resource utilization in each network slice. By doing so, it ensures fair distribution of resources among different services or slices, promoting uniform access to network capabilities for all users.

As we mentioned before, the project OREOS aims to achieve an end-to-end 5G network, with network slicing being one of its crucial components. The slicing management is typically done via ONAP, however, due to limited access to these resources, an alternative solution was pursued. Consequently, a fairness model was created, and a framework was developed to validate and implement this fairness model for network slicing management in 5G networks.

Since direct access to ONAP resources was not feasible due to resource consumption constraints, the focus shifted towards the slicing management part using the

developed framework. This framework involves key components such as NetworkX [NetworkX], MiniZinc [MiniZinc], and other complementary tools. By integrating these components seamlessly, the framework executes a sophisticated resource allocation process, ensuring a balance between service requirements and available network resources.

The Resource Allocation Framework's primary objective is to serve as a tool for simulating and optimizing resource allocation in network slicing scenarios. By allowing users to specify various parameters, the framework can effectively simulate real-life network configurations, making it versatile for network administrators and designers. This adaptability enables its application to a wide range of projects that require an intelligent and equitable resource allocation model.

Throughout the upcoming sections, we will delve into the intricate details of our framework, explaining its components, mathematical formulation, and optimization techniques. Additionally, we will demonstrate its applicability through diverse scenarios, showcasing how it significantly enhances resource allocation in modern networks. By the end of this section, readers will have a comprehensive understanding of the framework's potential and its importance in achieving efficient and fair network resource allocation.

6.1 Overview

The foundation of our resource allocation framework begins with the simulation of a network using NetworkX, which grants us the capability to define all essential network characteristics, including nodes, edges, and the bandwidth, CPU, and memory requirements of each service. The library supports various graph data structures, facilitating the efficient representation of real-world networks.

With the network simulation established in NetworkX, we proceed to the resource allocation phase, where MiniZinc takes center stage. Leveraging MiniZinc, we craft a model that encapsulates the resource allocation problem, factoring in critical constraints such as available CPU and memory in each node, total bandwidth in each link, and the resource requests of each service. We also define the objective function, aimed at optimizing fairness and efficiency in resource allocation across network slices.

Upon configuring the MiniZinc model, we seamlessly transmit the network simulation data from NetworkX to MiniZinc. This data encompasses the network topology, service requirements, and other pertinent parameters. MiniZinc, in turn, employs its solvers to identify the optimal resource allocation, adhering to all constraints and optimizing the predefined objective function.

The outcomes of the resource allocation process are then seamlessly fed back into NetworkX, enabling us to visualize and analyze the distribution of resources throughout the network. NetworkX's integration with Matplotlib empowers us to generate highly customizable visualizations, significantly enhancing our understanding of the network's topology and resource distribution.

6.2 Components Used

This section introduces the key components that form the foundation of our Resource Allocation Framework for Network Slicing. Each component plays a critical role in enabling efficient resource allocation and analysis within the network slicing context. We have already introduced these components in Section 2, and here we will specifically discuss the reasons behind choosing NetworkX, MiniZinc, and Matplotlib for our framework. By understanding the functionality and integration of these components, we can better appreciate how they contribute to the optimization and orchestration of network resources in the context of network slicing.

6.2.1 NetworkX

The decision to utilize NetworkX as our graph analysis library stemmed from several critical factors. Firstly, NetworkX offers a comprehensive array of functionalities and algorithms, providing a rich toolkit for effectively analyzing network properties. This diverse feature set proved indispensable in addressing the complexity of our resource allocation problem and allowed us to explore and model real-world networks with precision.

Seamless integration with the Python ecosystem was another pivotal consideration in selecting NetworkX. Given that our framework was developed in Python, opting for NetworkX felt like a natural and seamless fit. Its smooth compatibility with other popular scientific libraries, such as NumPy and Matplotlib, facilitated the effortless execution of complex mathematical computations and the visualization of results. This seamless integration greatly streamlined our workflow, enhancing overall productivity throughout the entire development process.

To represent real-world networks efficiently, it was crucial to have a tool that could create the network topology. NetworkX emerged as the ideal choice as it offers various functions to generate different types of graphs, such as Erdős-Rényi graphs, Watts-Strogatz graphs, and Barabási-Albert (BA) graphs. Since we wanted to create a topology based on the BA model, which will be elaborated further in this document in section 6.4, NetworkX naturally provided the necessary functionality.

To achieve a network representation as close to reality as possible, we needed to assign attributes to nodes and edges, simulating routers or links in a genuine network. NetworkX enables us to precisely set attributes, resulting in more detailed and informative representations of networks.

Lastly, to gain a visual understanding of our topology and better comprehend our network, we required a reliable visualization tool. Here again, NetworkX demonstrated its utility by seamlessly integrating with the Matplotlib library, allowing us to visualize networks and graph structures using customizable plotting functions. This visualization capability proved invaluable in comprehending network topology and relationships effectively.

6.2.2 MiniZinc

To mathematically represent the resource allocation problem within the network slicing context, we rely on MiniZinc, a powerful constraint modeling language and solver interface tailored for tackling combinatorial optimization and constraint satisfaction problems.

The inclusion of MiniZinc in our framework serves a crucial purpose – to validate the fairness model specifically developed to address optimization problems like allocating resources (e.g., bandwidth, processing power) optimally across different network slices while adhering to their unique requirements and constraints. This model has been meticulously designed to ensure equitable and efficient resource distribution among slices, contributing to the overall performance enhancement of network slicing.

Furthermore, MiniZinc provided excellent support for integrating with other programming languages, including Python, which was the language used to develop the rest of our framework. This allowed us to seamlessly incorporate MiniZinc models and solutions into our Python-based framework, enabling us to leverage the strengths of both MiniZinc and Python for a more robust and flexible solution.

The primary reason behind selecting Minizinc as our platform of choice is its seamless integration with ONAP. Minizinc's solver-independent approach allows for effortless incorporation of the optimization model into the ONAP environment. Leveraging Minizinc's interfaces enables us to adapt the model to ONAP's specific requirements, making it a perfect fit for our optimization solution within the broader network automation and orchestration provided by ONAP.

In particular, the ONAP Optimization Framework (OOF) adopts Minizinc as its constraint modeling language, providing a standardized and well-established foundation for optimization within the ONAP ecosystem. This decision ensures that we can capitalize on the integration efforts and advantages offered by this robust framework, streamlining our optimization implementation within ONAP. [ONAP Architecture]

6.2.3 Matplotlib

Matplotlib, a widely used Python plotting library, plays a crucial role in our Resource Allocation Framework for Network Slicing. As an essential component, Matplotlib enables us to visualize and interpret the solutions generated by our resource allocation model, providing valuable insights into the allocation of network resources across different slices.

One of the key reasons for choosing Matplotlib is its flexibility and versatility in creating a wide range of high-quality plots and visualizations. This capability empowers us to represent complex network structures, resource allocations, and performance metrics in a clear and intuitive manner. By leveraging Matplotlib's extensive set of visualization tools, our framework allows network operators and researchers to gain a comprehensive understanding of the resource allocation out-

comes.

The seamless integration of Matplotlib with NetworkX further enhances the visual representation of network topologies, as NetworkX serves as the underlying graph analysis library in our framework. This integration enables us to transform NetworkX graph data structures into visually appealing and informative plots, showcasing the allocation of services, links, and nodes across different network slices.

6.3 Inputs for Fairness Model

In this section, we delve into the essential input data that drives our resource allocation framework for network slicing. The framework operates based on input files that define the specific network scenario, encompassing various parameters crucial to the allocation process. These input files serve as the backbone of our framework, providing a structured representation of the network topology, service requirements, link attributes, resource constraints, and other critical elements.

Let's explore the key components of the input data:

- **Network Topology Representation:**

  The input data comprehensively describes the network topology using graph-based representations. Nodes and links are defined, where nodes represent network elements like routers, switches, or servers, and links symbolize the connections between these nodes.

- **Service Requirements:**

  The input data includes the service requirements for each individual service to be allocated in the network. These requirements consist of key parameters such as the amount of bandwidth required by each service to function optimally, the CPU processing power needed by each service to execute its tasks, and the memory capacity required by each service to store and process data.

- **Link Attributes:**

  The input data incorporates attributes of the links between nodes. These attributes represent the characteristics of the network links and influence the service allocation decisions. These attributes are the available Bandwidth, which reflects the total available bandwidth in each link, representing the capacity of data transmission, and the delay, which represents the delay or latency in transmitting data between nodes connected by the link.

- **Resource Constraints:**

  The input data may include resource constraints, such as the total CPU and memory available in each node. These constraints restrict the allocation of services to ensure that the nodes do not become overloaded.

- **Slice and Service Parameters:**

  The input data also contains parameters related to network slices and services, such as slice assignments which indicate which services belong to which network slices, guiding the framework in achieving slicing-specific objectives, and Link and node allocations within slices, which determine which links and nodes are available for each network slice, further influencing service placement.

- **Weight Parameter (W):**

  The input data includes the weight parameter used in the objective function. This parameter represents the penalization factor that influences the allocation of services and links based on their slice assignment and usage. It enables fairness in resource allocation across network slices.

All these input parameters adhere to a structured format determined by the array index. To provide a clearer understanding, in arrays containing slice attributes, such as bandwidth and latency, the initial element corresponds to the first slice, and this ordering persists for all subsequent elements.

This pattern is visually depicted in Figure 6.1. For instance, in this illustration, the variable *bw_link* contains the bandwidth values of the links. Consequently, the initial value of 5000 corresponds to links belonging to the first slice; 10000 is assigned to links in the second slice; and 50000 designates the bandwidth values for links within the third slice.

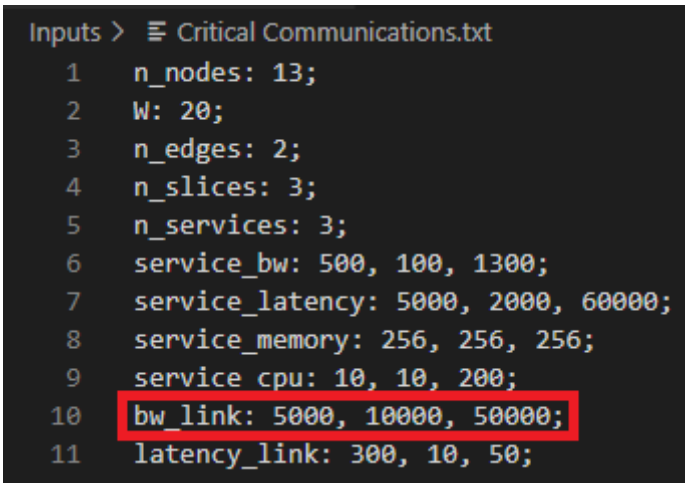


Figure 6.1: Input Example

By representing the network topology, service requirements, link attributes, and resource constraints in a structured and accessible format, the input data serves as the foundation of our framework. It allows network administrators, researchers, and operators to efficiently define and simulate various network scenarios, ensuring that the resource allocation model provides optimal and well-informed solutions for network slicing.

6.4 Network Topology

In the realm of computer networks, a fundamental aspect that lays the foundation for efficient data transmission, resource allocation, and overall network performance is "Topology." In its essence, network topology defines the physical or logical arrangement of interconnected nodes, devices, and communication links that form a network. Just as the layout of roads and intersections dictates how traffic flows in a city, network topology governs how data packets traverse through interconnected nodes in a computer network.

In this context, there are two primary types of network topology: physical and logical. Physical topology refers to the actual physical layout of network components, including devices like computers, routers, switches, and the cables that interconnect them. On the other hand, logical topology abstracts the physical layout and focuses on how data flows and communicates between nodes, independent of the physical medium.

Various topologies can be employed to design computer networks, each with its strengths and weaknesses. Some common network topologies include the star topology, bus topology, ring topology, mesh topology, and tree topology, among others. Each topology configuration caters to different network requirements, such as scalability, fault tolerance, and cost-effectiveness. In our framework, we generate the network topology using NetworkX with the Barabasi-Albert model.

Barabási-Albert graph

In our research framework, the strategic utilization of the BA graph model holds paramount importance, serving as a crucial tool for generating synthetic networks and enabling rigorous testing and simulation across diverse scenarios.

Out of the many network generation models available, we chose the BA model because it can create networks that resemble real-world networks with scale-free properties. Scale-free characteristics are commonly found in many real networks, making the BA model a suitable choice for our research. The information related to this type of topology was taken into account in the article [Barabasi-Albert model].

First and foremost, our approach to network synthesis prioritizes realism. The BA model is able to create networks that closely resemble real-world systems found in various domains, such as social networks, citation networks, and transportation networks. By capturing these realistic features, the synthetic networks we generate provide a strong basis for in-depth analysis and valuable insights.

Another key aspect is the **power-law degree distribution** seen in BA graphs, which mirrors the connectivity patterns seen in real networks. This distribution means that a few nodes have many connections, while most nodes have fewer connections. This reflects how connectivity works in many real networks, where some entities have far more connections than others.

Furthermore, the **scale-free nature** inherent in BA graphs contributes to the realism of our synthetic networks. This characteristic is characterized by a heteroge-

neous distribution of node degrees, where a small number of highly connected hubs coexist with the majority of nodes having limited connections. These critical hubs play a pivotal role in shaping the overall connectivity and resilience of the network. On the other hand, most nodes in the network have relatively few connections.

We can see a representation of the difference between a scale-free graph and a random network in Figure 6.2. In a scale-free network, we observe a few nodes with a significantly higher number of connections, forming hubs, while the rest of the nodes have only a few connections. This results in an uneven degree distribution and a network structure that mirrors real-world systems, where certain entities have a disproportionate number of connections compared to others. On the contrary, a random network lacks this heterogeneous degree distribution and exhibits a more uniform connectivity pattern, where nodes have similar degrees.

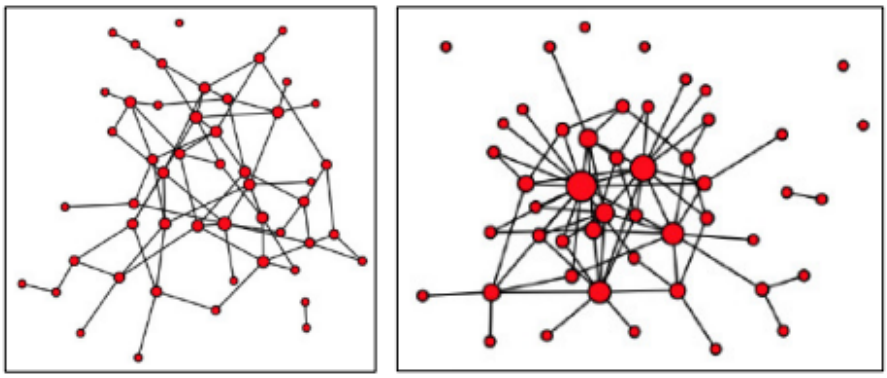


Figure 6.2: Random Network vs Scale-Free Network [Scale-Free Networks]

Lastly, the preferential attachment mechanism in the BA model aligns well with how many real networks grow over time. This mechanism is similar to scenarios seen in social media networks, where newly added entities prefer to link with well-connected nodes rather than less popular ones. This preference leads to a positive feedback loop, magnifying differences and influencing how the network evolves.

The BA model can represent this scenario of social media networks, where well-known influencers attract more new followers due to their popularity. As a result, these influencers end up having a disproportionate number of connections compared to other users, forming hubs in the network. This phenomenon creates a scale-free network structure with a few highly connected nodes and many nodes with limited connections.

We conducted a comprehensive evaluation of various graph generation models, including the Erdős-Rényi and Watts-Strogatz models, in conjunction with the Barabási-Albert (BA) model. Our analysis unveiled that the Erdős-Rényi and Watts-Strogatz models typically yield networks characterized by a more uniform degree distribution. However, this uniformity falls short of capturing the heterogeneous connectivity patterns observed in many real-world networks. These models rely on random processes and may not effectively replicate the dynamic growth processes that underlie real-world network evolution. On the other hand,

scale-free networks generated by the BA model exhibit increased resilience to random node failures when compared to those produced by Erdős-Rényi or Watts-Strogatz models.

The choice of the BA model seamlessly aligns with scenarios where the objective is to examine networks with highly influential or well-connected nodes, a common feature in many real-world systems. In contrast, Erdős-Rényi and Watts-Strogatz models are better suited for situations where network nodes display more uniform degrees and connectivity patterns.

The generation of the Network Topology Process

The network is composed of several nodes and links. Nodes in the network represent the individual components or entities within the infrastructure, such as routers, switches, or servers. The creation of nodes is based on the input data, which provides details about the physical layout of the network. Each node is uniquely identified, and its attributes, such as processing capacity (CPU) and memory, are defined.

The Barabasi-Albert model starts with a small number of nodes *(m)* and gradually adds new nodes one by one, each connected to m existing nodes. This process continues until the desired number of nodes *(n)* is reached

Edges in the network represent the connections between nodes, indicating the existence of links between them. These links serve as communication channels through which data can flow between nodes. The creation of edges is influenced by the *q_link* matrix.

For each new node, we add *m* edges that connect the node to m existing nodes. The probability of a new node connecting to an existing node is proportional to the existing node's degree (number of neighbors). Nodes with higher degrees are more likely to attract new connections, reflecting the "rich-get-richer" principle.

The *q_link* matrix is a binary matrix that indicates the links between nodes in the network. It has a size of N x N, where N is the number of nodes in the network. Each entry (i, j) in the *q_link* matrix is either 0 or 1, representing the absence (0) or presence (1) of a link between nodes *i* and *j*.

The process of creating edges based on the *q_link* matrix involves examining each entry in the matrix. For each entry (i, j) with a value of 1, an edge is created between node *i* and node *j*, indicating that a link exists between them. Conversely, for entries with a value of 0, no edge is created, indicating that there is no direct link between the corresponding nodes.

To gain a clearer insight into the functioning of this generation process, refer to Figure 6.3, which illustrates the structure of the *q_link* matrix. Within this matrix, each array corresponds to a specific slice; in this case, we have three slices. If an edge is denoted with a value of 1 in the corresponding array, it signifies that the edge belongs to the respective slice. For example, in slice 1, the links (0,3) and (3,0) are included, while in slice 2, the links (2,3), (3,2), (3,4), and (4,3) are designated.

```
q_link: [
[[0,0,0,1,0],
 [0,0,0,0,0],
 [0,0,0,0,0],
 [1,0,0,0,0],
 [0,0,0,0,0]],

[[0,0,0,0,0],
 [0,0,0,0,0],
 [0,0,0,1,0],
 [0,0,1,0,1],
 [0,0,0,1,0]],

 [[0,0,0,0,0],
  [0,0,0,0,0],
  [0,0,0,0,0],
  [0,0,0,0,0],
  [0,0,0,0,0]]
];
```

Figure 6.3: q_link matrix

6.5 MiniZinc Model

In this section, we present the implementation of the mathematical model presented in section 5, using MiniZinc as a language to optimize the fairness of bandwidth and CPU allocation in multi-slice networks.

The primary objective of this model is to maximize the fairness of resource allocation for each slice while ensuring an equitable distribution of overall resources across the network. To achieve this, we carefully translate the mathematical model into MiniZinc syntax.

Our validation process utilizes MiniZinc to test and assess the performance of the Fairness Model. We employ various metrics to evaluate the fairness and efficiency of resource allocation.

As we mentioned before, the model implemented in the MiniZinc platform was based on the fairness model presented in Section 5. Our role in this phase of the research was to bring the mathematical model to life by transforming abstract mathematical equations into practical constraints and conditions that could be executed and evaluated using MiniZinc.

Solving the Model using the Gecode Solver

As Python served as the primary language for the rest of our resource allocation framework, we greatly valued Gecode's interoperability with Python. This compatibility allowed us to effortlessly incorporate the solver's solutions into our Python-based framework, empowering us with a rich set of tools and libraries for various tasks.

By leveraging Gecode's efficiency, constraint programming expertise, seamless integration with MiniZinc, and interoperability with Python, we were able to create a robust and highly effective resource allocation framework for network slicing.

Process of solving the Minizinc Model

The MiniZinc model, written in a high-level declarative language, needs to be compiled into a format that the Gecode solver can understand and process. With the model compiled, the Gecode solver is selected as the solver of choice. This is represented in figure 6.4.

```
input = read_input("Inputs/Critical Communications.txt")

data = topology(input)

model = minizinc.Model("model.mzn")
gecode = minizinc.Solver.lookup("gecode")
inst = minizinc.Instance(gecode, model)
```

Figure 6.4: Gecode implementation

The Gecode solver proceeds to search for a feasible solution that satisfies all the constraints defined in the model. During this process, the Gecode solver keeps track of the best solution found so far. As it explores the solution space, if it encounters a solution that improves upon the current best, it updates the best solution accordingly. This way, the solver continues to refine its search, striving to find the best possible solution.

Once the Gecode solver completes its search, it reports the results. If a feasible solution that satisfies all constraints is found, it is presented as the output. The solution includes the allocation of resources, such as CPU, memory, and bandwidth, for each service in the network slicing scenario. If an optimal solution is reached, it represents the most efficient allocation that maximizes fairness and meets all the defined constraints.

After obtaining the solution, it is further analyzed to evaluate its effectiveness and fairness in resource allocation. Various performance metrics and visualizations, facilitated by the integration with Matplotlib, provide insights into the network's resource utilization and the impact of the resource allocation on different slices and services.

6.6 Summary

In the context of modern networking, the efficient allocation of network resources is crucial due to increasing demands for diverse services. This chapter introduces a comprehensive framework designed to optimize network resource allocation while prioritizing fairness. While various methods and models exist for resource allocation, this framework takes a holistic approach, considering overall resource

utilization across network slices. The objective is to ensure equitable distribution of resources among services and users.

The framework is particularly significant for the OREOS project, which aims to create an end-to-end 5G network with network slicing as a key component. Given limited access to resources like the ONAP orchestrator, an alternative approach was pursued. A fairness model was developed, and a framework was created to validate and implement this model for network slicing management in 5G networks. By utilizing components like NetworkX, MiniZinc, and Matplotlib, the framework orchestrates a sophisticated resource allocation process, aligning service requirements with available resources.

A notable feature of the framework is its code availability in a repository, allowing others to access and contribute to its development. The careful selection of NetworkX and MiniZinc as core components ensured compatibility, enabling the generated data to be directly input into ONAP. This compatibility allows for a smoother transition from the simulation framework to the actual ONAP environment, enhancing the practicality and applicability of the resource allocation model.

The framework's foundation lies in the generation of network topology using the Barabasi-Albert model, which creates synthetic networks mirroring real-world systems. NetworkX is used for network simulation, enabling efficient visualization and analysis. MiniZinc, a constraint modeling language, plays a pivotal role in crafting an optimization model that ensures fair resource allocation. The process involves translating mathematical models into MiniZinc code and utilizing the Gecode solver to find optimal or near-optimal solutions. The outputs are analyzed using various metrics and visualizations to assess the fairness and efficiency of the resource allocation.

The framework's adaptability and ability to simulate real-world scenarios make it a powerful tool for network administrators and designers. By understanding the components, mathematical formulation, and optimization techniques of the framework, readers can appreciate its potential for enhancing resource allocation in modern networks while ensuring fairness.

Chapter 7

Model Validation and Testing

This chapter presents a comprehensive methodology for validating and testing the network resource allocation model in multi-slice networks. The validation process is conducted in progressive stages, starting with small-scale tests and gradually progressing to more complex scenarios. This approach allows for a thorough evaluation of the model's performance and behavior under varying conditions.

Initially, the validation involves generating small networks, enabling manual calculation of expected results, which are then compared with the model's outputs to verify resource allocations' correctness. As confidence in the model's accuracy grows, more challenging tests with larger networks and multiple services are conducted, using random values for service resource demands and network link characteristics. This step ensures that the model performs effectively under diverse scenarios and varying resource demands.

A critical aspect of the validation process involves evaluating the model's scalability to handle larger networks without compromising performance. This ensures that the model is applicable in real-world scenarios with high resource demands and large-scale networks.

Throughout the validation process, the model's results are meticulously analyzed, ensuring that it meets the requirements of fairness and efficiency in resource allocation among network slices. Performance metrics are utilized to support the conclusions about the model's effectiveness and applicability in practical scenarios.

It is important to note that we are not using a heuristic to validate the model, and the validation does not focus on the quality of the solution but rather on verifying whether the solution is correct in terms of meeting the requirements and objectives of the problem. The primary objective of validation is to guarantee that the model consistently delivers precise and consistent results that align with the distinct requirements of diverse network scenarios. This involves assessing whether, within each designed scenario, the model adeptly chooses the links and nodes that best contribute to minimizing the final outcome, all while considering the unique demands imposed by these require-

ments.

In essence, the validation process confirms that the model reliably provides correct solutions by effectively allocating services while complying with the constraints imposed by both the test cases and the model itself. This meticulous validation ensures the model's reliability and applicability in practical scenarios, where accurate resource allocation is crucial for optimizing network performance and resource utilization.

7.1 Validation Methodology

The validation methodology was developed in progressive stages, starting with small-scale tests and gradually progressing to more complex scenarios to evaluate the model's performance and behavior. Initially, we generated small networks, allowing us to manually calculate the expected results. We compared these results with those obtained from the model to verify the correctness of resource allocations.

As we gained confidence in the model's accuracy, we advanced to more challenging tests involving larger networks and more services. In these cases, we used random values for service resource demands and network link characteristics. While these values did not have specific justifications, they were randomly selected to assess how the model performed under varied conditions.

Another important aspect was evaluating the model's scalability. We tested its ability to handle increasingly larger networks, ensuring that it maintained acceptable performance even in large-scale scenarios. This was essential to confirm that the model could be applied in real-world environments with large-scale networks and high resource demands.

Additionally, we conducted sensitivity tests to analyze the impact of different parameters and input values on the final results. We adjusted weight values, resource capacities, and service requirements to understand how these changes would affect resource allocations and fairness among the slices.

A significant part of the validation involved comparing the results with benchmarks established by other resource allocation methods in multi-slice networks. This allowed us to assess the model's efficiency and relative performance compared to existing approaches, demonstrating its effectiveness and highlighting its advantages.

Throughout the validation process, we carefully analyzed the results, ensuring that the model met the requirements of fairness and efficiency in resource allocation among network slices. Performance metrics and comparisons with traditional approaches were used to support our conclusions about the model's effectiveness and applicability in practical scenarios.

7.2 Performance Metrics

In this section, we outline the performance metrics used to evaluate the effectiveness and efficiency of our resource allocation model in multi-slice networks. These metrics play a crucial role in assessing the model's performance and its ability to achieve fair resource allocation while optimizing network utilization. By quantitatively measuring various aspects of the model's performance, we gain valuable insights into its behavior and suitability for real-world scenarios.

7.2.1 Fairness Metric

The fairness metric in the network resource allocation model aims to evaluate the equitable distribution of resources (bandwidth and CPU) among the different slices in a multi-slice network. The primary objective is to achieve a balanced and fair allocation of resources, ensuring that no single slice is significantly disadvantaged in terms of resource availability.

In the framework of the objective function outlined in Section 5, and considering the resultant variable Z derived from the model, the fairness metric evaluates the efficacy of resource allocation among various slices. This assessment is made by considering the cumulative utilization of resources within each slice. The Z variable serves as a reflection of the extent to which the fairness objective is achieved in the resource allocation process. By minimizing the Z variable, the optimization process seeks to align the allocation of resources in a manner that maximizes fairness across slices, while abiding by the specified constraints.

In this optimization framework, a high value of Z suggests that the allocation of resources across slices is facing challenges in meeting the fairness and resource allocation constraints. This indicates that the resource distribution might be disproportionate, potentially favoring certain slices over others. In other words, a **high Z value** reflects an allocation scenario where the **fairness objective is not well achieved**, and there may be slices that are significantly disadvantaged in terms of resource availability.

On the other hand, a **low value of Z** indicates that the allocation of resources is in **closer alignment with the fairness and resource allocation constraints**. In this scenario, **resources are more evenly and fairly distributed among different slices**, and the optimization process is successful in achieving its objectives. A low Z value signifies that the **allocation strategy is effective** ensuring that each slice receives its due share of resources, minimizing any discrepancies or imbalances.

By optimizing the fairness metric, the network resource allocation model aims to mitigate potential performance bottlenecks and ensure that slices with high resource demands receive adequate allocations without impacting other slices negatively. This balanced distribution of resources leads to improved service quality, reduced congestion, minimized service disruptions, and ultimately, a higher level of user satisfaction across the entire network.

7.2.2 Slice Isolation

In our network resource allocation model for multi-slice networks, the Slice Isolation metric holds significant importance in preserving the integrity and independence of individual slices. Its primary goal is to ensure that services are exclusively allocated to their designated slices and not spread across other slices, except when dictated by resource constraints.

To achieve this isolation, we rely on a crucial element within our model known as the penalization parameter, represented as W in Equation 5.1. Whenever the model attempts to allocate a service to a slice other than its designated one, a penalization factor comes into play. This penalization acts as a deterrent, discouraging the model from making inappropriate allocations and reinforcing strict adherence to slice boundaries. In cases where a service is correctly allocated to its designated slice, the penalization W remains dormant, as it aligns with the desired behavior.

While preserving slice isolation is essential for maintaining the integrity of network slices, we must also strike a balance with resource efficiency. In certain scenarios, resource constraints or exceptional circumstances may necessitate allocating a service to a different slice. In such situations, the penalization is applied, but the allocation may proceed if no other viable options are available.

The incorporation of the penalization parameter W in the model encourages a thoughtful evaluation of resource availability and the consequences of making cross-slice allocations. It motivates the model to prioritize resource allocation within the designated slice, while still allowing for flexibility in cases where resource constraints demand alternative allocations.

To calculate the slice isolation metric, we adjust the value of the W variable to be either more strict or relaxed. A **higher value of W instructs the model to make every effort to prevent the service from being implemented in another slice that does not belong to it.** Conversely, a **lower value gives the model more flexibility to allocate the service in the most optimal manner, even if it means crossing slice boundaries.** This flexibility allows us to strike a balance between maintaining strict slice isolation and optimizing resource utilization based on the specific requirements of the network scenario.

7.3 Datasets and Inputs

Slice Values Methodology

As previously discussed, our validation methodology was structured into three distinct phases:

- **Initial Tests:** These involved creating small networks to enable manual calculation of expected outcomes. This initial phase laid the foundation for our validation process.

- **Intermediate Tests:** These tests introduced increasing complexity and encompassed a range of random scenarios. This phase aimed to explore the model's behavior under diverse conditions with growing intricacy.
- **Real Scenario Tests:** The final phase focused on the practical application of our model in real-world environments. This step aimed to validate the model's effectiveness when subjected to actual usage scenarios.

In our model, we considered three types of network slices: enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and Massive Machine-Type Communications (mMTC). The key parameters we used from these slices were bandwidth and latency, which played a crucial role in resource allocation.

The bandwidth and latency values that we assigned to each slice are represented in Table 7.1:

Network Slice	**Latency (ms)**	**Bandwidth (Kbps)**
eMBB Slice	50	50000
URLLC Slice	10	10000
mMTC Slice	300	5000

Table 7.1: Latency and Bandwidth Values for Network Slices

The values of latency were chosen based on the dataset from [5G Dataset Network Slicing CRAWDAD Shared], providing realistic latency requirements for each slice. Regarding the bandwidth values, we used the values based on [Network slicing for eMBB, URLLC, and mMTC: An uplink rate-splitting multiple access approach].

The provided details pertain exclusively to intermediate and real-test scenarios, aligning with our aim to acquire authentic data for testing the fairness model. For the initial test cases, specific values were adopted. In these cases, the selection of bandwidth values for the links was guided by insights from the research paper [SALEM: Service Fairness in Wireless Mesh Environments]. Accordingly, wireless connections were attributed a bandwidth of 54 Mbps, while fiber optic connections were set at 1000 Mbps. This choice was made because the initial tests involved scenarios with only one network slice.

Node values Methodology

In our network topology representation, we modeled a combination of virtual machines (VMs) using NetworkX to represent different nodes, and we incorporated two distinct hardware configurations.

- The first configuration involved professional hardware with higher resource consumption;
- The second configuration utilized commodity hardware with lower resource consumption.

Additionally, we included a dedicated node to represent the cloud, which had the most abundant resources compared to the other nodes. The values used to categorize each node were based on the dataset [Thantharate et al., 2022].

Specifically, 50% of nodes had values from VMs with "op ccc" values, which stood for On-Premises Commodity Cloud Computing. To simulate these cloud computing nodes we used professional hardware like Cisco IR829 Industrial Integrated Services Routers.

The remaining 50% of the nodes were assigned to VMs with the "ccc to cloud" values, representing commodity cloud computing. This approach involved using commodity hardware, such as Raspberry Pi devices.

Furthermore, we designated a single dedicated node to represent the cloud, which possessed more resources than the other nodes. To represent this cloud node we used a High-Performance Cloud Instance.

The characteristics of the devices used for simulation are tabulated in Table 7.3. This table provides an overview of the CPU and RAM specifications for each device. The units used to represent these values are Million Instructions per Second (MIPS) for CPU and Megabytes for RAM.

Device	CPU (MIPS)	RAM (MB)
Cisco IR829 Industrial Integrated Services Routers	16000	8000
Raspberry Pi	9600	4000
High-Performance Cloud Instance	30000000	131072

Table 7.2: Cloud Device Characteristics

In a manner akin to the method used for selecting bandwidth and latency values, the determination of CPU and memory values for distinct nodes in the initial and intermediate test cases adhered to a deliberate strategy. This strategy mirrored the categorization scheme employed in real-test scenarios, wherein nodes were grouped into three distinct sets: comparable to commodity nodes, professional-grade nodes, and cloud nodes.

Device	CPU and RAM
Lower CPU and memory (Commodity)	Random values ranging from 1 to 10
Intermediate CPU and memory (Professional)	Random values ranging from 10 to 20
Elevated CPU and memory values (Cloud)	Random values ranging from 20 to 300

Table 7.3: Initial and Intermediate Node Characteristics

Services values Methodology

In the framework of the initial and intermediate test cases, we employed a randomized generation process to define parameter values for various services. In contrast, when striving to authentically emulate services within real-scenario test cases for our model, we embraced a comprehensive approach. This approach took into consideration a spectrum of parameters, spanning critical elements like bandwidth, latency, CPU, and memory requirements for each distinct service.

The CPU and memory values utilized in our test cases were carefully extracted from real-world scenarios documented in the article [Service Function Chaining in Wildfire Scenarios]. Leveraging these values provided a robust framework for consistently simulating resource allocation across diverse scenarios. To ensure an accurate representation of the resource requirements for each service, we categorized them based on their functions and cross-referenced them with the corresponding CPU and memory values from the aforementioned article.

The categorization process entailed the selection of the suitable service function for each implemented service. We achieved this by referencing the table located in Chapter A, enabling us to confidently match each service with its most fitting function. This systematic approach ensured a precise representation of the resource requirements for various services, thereby enhancing the fidelity of our simulation to real-world conditions.

For determining suitable bandwidth and latency values, we referred to this article to align the characteristics of each service with appropriate network slice specifications.

Scenario 1: Network Slicing for Critical Communications in Shared 5G Infrastructures

Service	**Slice**	**Bandwidth (Kbps)**	**Latency (ms)**	**CPU (MIPS)**	**Memory (MB)**
Smart Metering	mMTC	500	5000	10	256
Floating Car	uRLLC	100	2000	10	256
Multimedia	eMBB	1300	60000	200	256

Table 7.4: Test Case I: Service Requirements

Smart Metering involves collecting data from utility meters, such as electricity, water, or gas meters, using IoT devices. So we categorized the function of this service as "IoT data acquisition."

Floating Car Data typically involves collecting and analyzing real-time vehicle location information using GPS location data. For this service, we chose the function "GPS location data."

Lastly, Multimedia represents video transmission, so we labeled its function as "video transmission."

Scenario 2: Optimizing Fairness in Wireless Mesh Networks

Real-Time Virtualization Technology (RT-VT) falls into the IoT data processing category, as it involves virtualizing real-time systems and executing real-time processes.

Voice traffic belongs to the Conf. voice channels category, as it entails audio data transmission for voice calls and real-time communications.

Burst-user-driven (BUD) traffic is classified under the IoT data acquisition cat-

Service	Slice	Bandwidth (Kbps)	Latency (ms)	CPU (MIPS)	Memory (MB)
RT-VT	mMTC	1500	5000	20	256
Voice traffic	uRLLC	24.4	30000	30	256
BUD traffic	eMBB	12000	5000	10	256

Table 7.5: Test Case II: Service Requirements

egory, encompassing periodic and bursty data transmissions from sensors, IoT devices, and M2M connectivity.

Scenario 3: OREOS Smart city

Service	Slice	Bandwidth (Kbps)	Latency (ms)	CPU (MIPS)	Memory (Mb)
Pedestrian safety	mMTC	1500	120000	700	256
Air quality monitoring	uRLLC	500	5000	10	256
Crime prevention	eMBB	1500	120000	800	256

Table 7.6: Test case III: Service Requirements

Pedestrian safety is triggered by the event of a citizen crossing a road, in a local covered by video surveillance cameras. The initial step is the capture of video and transfer of such information for analysis. Based on the information provided in the previously referenced article, we have categorized this service under the Service Function: Video real-time analysis. This critical function plays an indispensable role in promptly detecting pedestrians crossing the road in real-time. By analyzing the video feed from surveillance cameras, potential safety concerns can be identified and addressed promptly.

Air quality monitoring is triggered with the deployment of IoT devices in several city locations and the necessary configuration of the data pipeline for collection, aggregation, and processing. We have categorized this service under the function: of Data acquisition from biosensors, and sensors. The acquisition of data from IoT devices, such as biosensors and sensors, assumes paramount importance in gathering comprehensive air quality data from diverse city locations. This entails the systematic gathering of data from the deployed IoT devices to facilitate robust air quality monitoring.

Crime prevention is triggered by the detection of an abnormal event regarding a citizen covered by video surveillance. Such abnormal event (such as a fall or robbery) is detected by video cameras. The story can also be triggered by the sharing of information from a tourist/citizen regarding the perception of safety in a certain place. Such feedback is performed through the “Citizen App” made

available by the municipality. For this service, we have categorized the function as Video real-time analysis. Video real-time analysis plays a pivotal role in crime prevention, empowering surveillance systems to promptly detect abnormal events, such as falls or robberies, in real-time, based on the video feed from cameras.

Bandwidth and CPU Normalization

In complex scenarios, normalizing bandwidth and CPU values becomes crucial. Normalization, a mathematical technique, rescales values to a standardized interval, often from 1 to 2 in our case. This choice avoids nullifying computational impact as with a 0-1 range, where the minimum value is always 0, rendering calculations meaningless. By opting for 1-2, meaningful calculations occur, ensuring effective computation.

Normalization's purpose is to harmonize values on a common scale for easier comparison and analysis. This aids optimization problems with varied variable magnitudes. Normalization counters variable dominance due to numerical range; without it, larger values could disproportionately influence optimization. For example, varying bandwidth (100-10,000 units) versus CPU (1-10 units) requirements might incline optimization towards larger bandwidth. Normalization aligns these requirements to foster equitable optimization influence.

7.4 Test Results and Analysis

This section delves into the outcomes of the conducted tests and provides a thorough analysis of the obtained results. For enhanced clarity, distinct colors will be employed to visually distinguish the paths taken by each service. The calculation of the fairness metric, denoted as Z, involves aggregating the utilized bandwidth along the paths and incorporating values related to service allocation.

The forthcoming test cases will be presented in a structured tabular format, encapsulating the following essential attributes:

- **Starting Node:** This indicates the origin point of the path where the journey commences.
- **End Node:** It designates the terminal destination where the path concludes.
- **Bandwidth Requirement:** This specifies the amount of bandwidth essential for the service to operate effectively.
- **Latency Constraint:** It signifies the maximum permissible time delay, a crucial consideration for the service's performance.
- **CPU Capacity:** This outlines the processing power required by the service, quantified in terms of CPU units.
- **Memory Allocation:** It delineates the extent of memory resources allocated to facilitate the service's operations.

7.4.1 Initial Tests: Small-Scale Verification and Manual Validation

In the upcoming tests, we will present a series of test cases aimed at validating our model. These test cases will employ a limited set of values to confirm the accuracy of our fairness model. In this initial phase, our focus is to gain a clear understanding of how the model selects which links to utilize. To facilitate our calculations and analysis, we are conducting these tests with just one network slice. Given this single-slice setup, we will not consider the slice isolation metric, as it cannot be practically applied in this preliminary stage. Consequently, our evaluation in this section will solely revolve around the fairness metric, denoted as variable Z, as we assess whether it produces the expected output results.

Test case 1

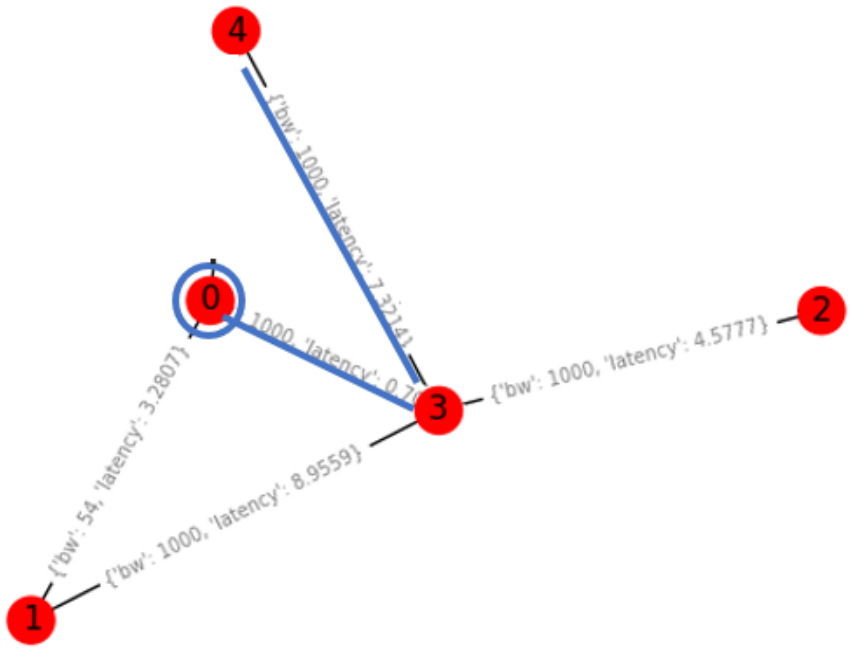


Figure 7.1: Graph Test Case I

Start Node	End Node	Band.	Latency	CPU	Memory
0	4	11.35060	10	10	10

Table 7.7: Test case I

Node	Memory (Gb)	CPU (Gb)
0	14.6292	10.8124
1	14.6292	10.8124
2	14.8852	16.2002
3	14.6292	10.8124
4	188.3523	52.4957

Table 7.8: Memory and CPU Specifications for Nodes

Upon analyzing the test case, we observe a path that begins at node 0, offering two potential destinations: link(0,3) and link(0,1).

If the path were to divert to link(0,1), it would use more bandwidth than necessary, as it would later need to use the link(1,3). This detour would result in an additional bandwidth usage of 11.3560 units compared to a direct route through link(0,3). However, once the path reaches node 3, the only available onward option is to use the link(3,4). Therefore, the model chooses that link leading to the optimal solution in this scenario.

To compute the fairness metric represented by z entails aggregating the utilized bandwidth along the path and the value denoting service allocation. In the context of this test case, the calculation is:

- 11.3560 (Service Bandwidth) * 2 (link(0,3) and link(3,4)) + 10 (Service required CPU)
- Z = 32.712

This result aligns precisely with the anticipated value, corroborating the accuracy of the model's calculations. As illustrated in Figure 7.1, the service allocation is observed to have been designated to Node 0. Notably, the service necessitates a CPU capacity of 10 units, a requirement that aligns with the available resources of Node 0, which boasts a total of 10.8124 units. Thus, Node 0 stands as a viable candidate for accommodating the service's computational demands.

Test case 2

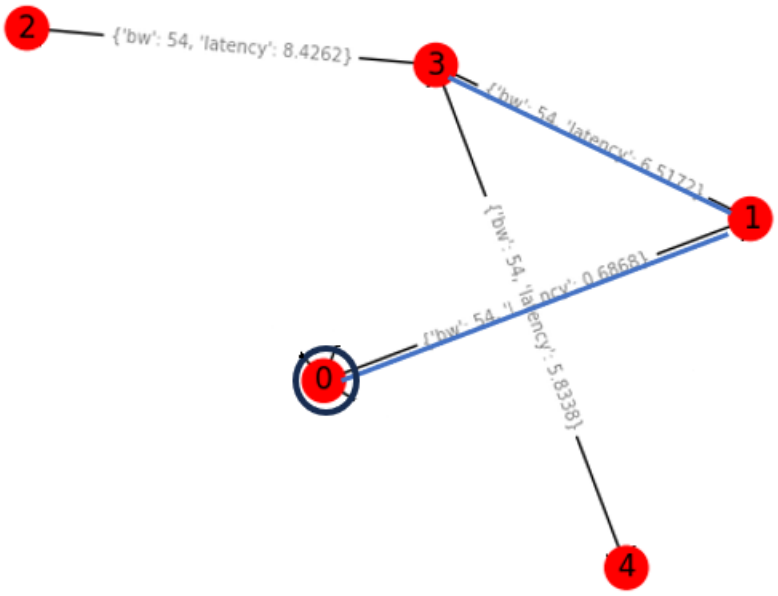


Figure 7.2: Graph Test Case II

Within this specific test case, a singular option presents itself: that of selecting link(0,1) followed by utilizing link(1,3) to complete the path.

The cumulative outcome of this choice is calculated as follows:

Start Node	End Node	Band.	Latency	CPU	Memory
0	3	0.3124	10	10	10

Table 7.9: Test case II

Node	Memory (Gb)	CPU (Gb)
0	12.7153	11.5524
1	61.6382	276.8038
2	13.082	12.1299
3	12.7153	11.5524
4	12.7153	11.5524

Table 7.10: Memory and CPU Specifications for Nodes

- 0.3142 (Service Bandwidth for link(0,1)) + 0.3142 (Service Bandwidth for the link(1,3)) + 10 (service-required CPU)
- Z = 10.6284

Significantly, this computed value aligns perfectly with the expected outcome, unequivocally validating the model's precise calculations. This outcome confirms that in this particular test case, fairness achieved the optimal solution, as the model adeptly selects the optimal links and nodes for resource allocation.

Test case 3

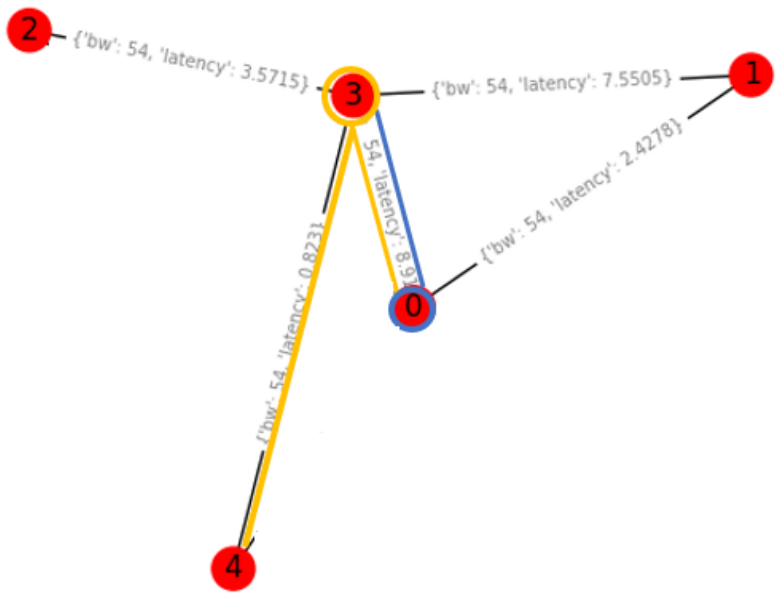


Figure 7.3: Graph Test Case III

In this particular test case, two distinct services are at play:

- **Service 1:** The model determines the optimal route by selecting the link(0,3). Although it could have opted for link(0,1) followed by link(1,3), this alternative trajectory would have unnecessarily consumed bandwidth. By electing

Service	Start Node	End Node	Band.	Lat.	CPU	Mem.	Color
1	0	3	10.0965	15	4	0	Blue
2	0	4	11	3	18	17	Yellow

Table 7.11: Test case III

Node	Memory (Gb)	CPU (Gb)
0	12.8949	14.0302
1	19.2668	11.8135
2	12.8949	14.0302
3	101.1108	282.8566
4	12.8949	14.0302

Table 7.12: Memory and CPU Specifications for Nodes

to traverse solely through the link(0,3), the model demonstrates a judicious decision that conserves bandwidth without compromise.

- **Service 2:** Selects the identical link (link(0,3)) as its optimal path. This route becomes the only viable choice to fulfill the requirement of culminating at node 4.

By analyzing the model's behavior in this specific test case, we can delve into the fairness metric, which demonstrated that the achieved fairness was indeed optimal. The outcome of this test case computation is as follows:

- **Service 1:** 11 (Service bandwidth) + 11 (Service bandwidth) + 18 (Service CPU)
- **Service 2:** 10.0965 (Service bandwidth) + 4 (Service CPU)
- $Z = 54.0965$

As for the allocation of resources, the first service finds its home in node 0, while the second service takes residence in node 3—a placement strategy that we previously elucidated.

Test case 4

Service Number	Start Node	End Node	Band.	Latency	CPU	Memory	Color
1	0	3	11.6138	13.2485	6.1743	17.274	Blue
2	0	4	2.8301	19.844	13.719	18.070	Yellow

Table 7.13: Test case IV

This test case holds significant intrigue in unraveling the model's behavior. In this illustrative instance, the apparent path of choice for the model seemed straightforward—opting for link(0,3) for Service 1 and traversing link(0,3) followed by

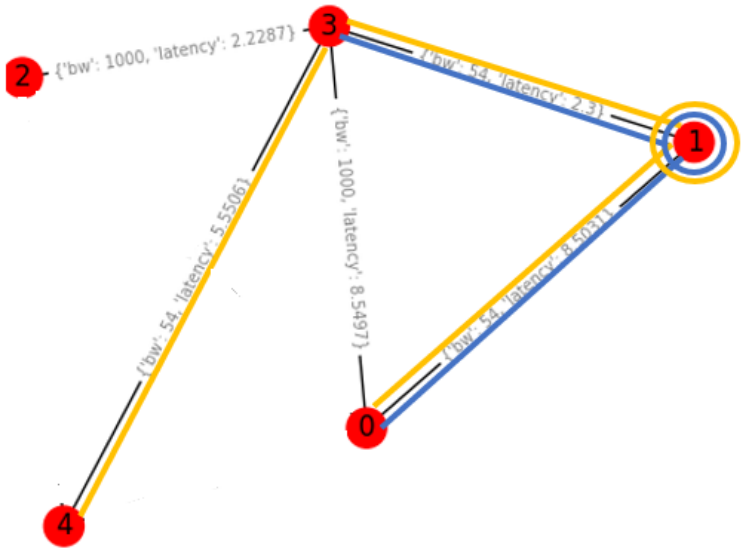


Figure 7.4: Graph Test Case IV

Node	Memory (Gb)	CPU (Gb)
0	13.888	14.8257
1	237.007	144.4957
2	13.888	14.8257
3	12.4904	17.6135
4	13.888	14.8257

Table 7.14: Memory and CPU Specifications for Nodes

link(3,4) for Service 2. However, a noteworthy twist comes to light: Node 1 emerges as the sole contender capable of accommodating the service requisites.

As a consequence, the model adeptly alters its trajectory. It strategically embarks upon link(0,1) to meet the resource prerequisites, subsequently resuming its journey through link(1,3). This shift effectively optimizes resource allocation, addressing the unique constraints posed by node 1's capabilities.

The strategic adaptations in path selection vividly showcase the model's aptitude for orchestrating resource allocation within the confines of network dynamics and constraints.

Consequently, the outcome of this test case unfolds as follows:

- For Service 1, the calculated value stands at (11.6138 * 2) for bandwidth, adding to 6.1743 for CPU, yielding a total of 29.4019.
- Service 2, on the other hand, contributes 2.8301 (bandwidth) multiplied by 3, accompanied by 13.719 for CPU, summing up to 22.1093.
- z = 51.5112

Test case 5

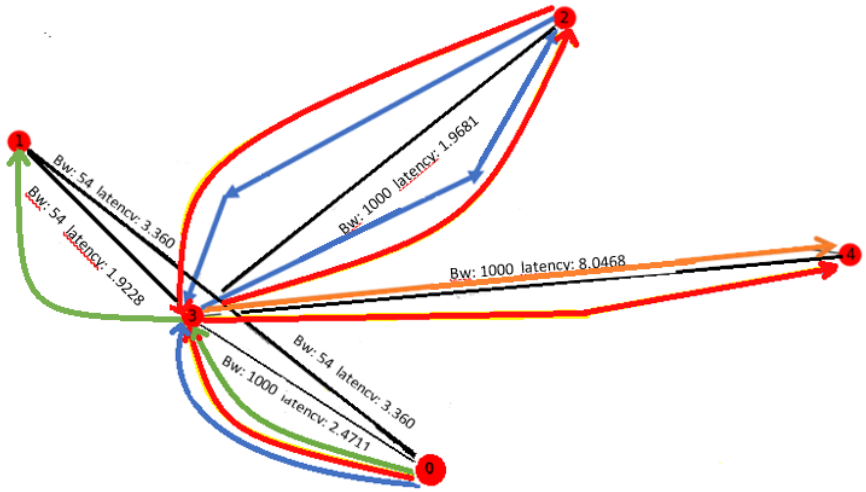


Figure 7.5: Graph Test Case IV

Service	Start Node	End Node	Band.	Latency	Memory	CPU	Color
1	0	3	0.8355	13.3811	0.4524	19.3813	Blue
2	0	4	5.9119	16.9656	8.0845	17.9326	Yellow
3	0	1	17.2739	7.5776	6.5545	11.3386	Green
4	3	4	11.022	8.0958	3.9704	1.4925	Orange

Table 7.15: Test case V

Service 1 once again struggles with a CPU allocation problem. As a resolution, it charts a unique course: veering towards node 2, the singular outpost capable of fulfilling its specific CPU prerequisites. With resources successfully allocated, Service 1 then gracefully returns to its intended path, converging once more at node 3—the designated terminus.

Consequently, the outcome of this test case unfolds as follows:

- For Service 1, the calculated value stands at (0.8355 * 3) for bandwidth, adding to 19.3813 for CPU, yielding a total of 21.8878.
- Service 2, on the other hand, contributes 5.9119 (bandwidth) multiplied by 4 (links used), accompanied by 17.9326 for CPU, summing up to 43.8422.
- Service 3, on the other hand, contributes 17.2739 (bandwidth) multiplied by 2 (links used), accompanied by 11.3386 for CPU, summing up to 45.8864.
- Service 4, on the other hand, contributes 11.022 (bandwidth), accompanied by 1.4925 for CPU, summing up to 12.5145.
- Giving a total Z of 123.1309.

Node	Memory (Gb)	CPU (Gb)
0	9.5511	8.1169
1	9.5511	8.1169
2	25.1739	157.6299
3	15.9234	17.9788
4	9.5511	8.1169

Table 7.16: Memory and CPU Specifications for Nodes

As for the allocation of resources, the first and second services find their home in node 2, while the third and fourth service takes residence in node 3.

Test case 6

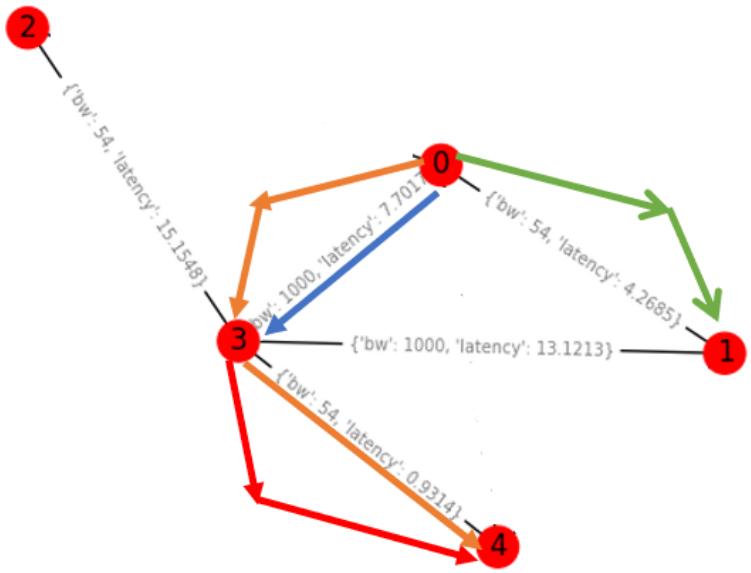


Figure 7.6: Graph Test Case VI

Service	Start Node	End Node	Band.	Latency	Memory	CPU	Color
1	0	3	19.7952	17.3856	0.288	16.0402	Blue
2	0	4	16.8884	12.7575	2.8488	9.333	Orange
3	0	1	2.4179	5.4979	7.7209	13.6389	Green
4	3	4	3.0744	5.6482	0.249	5.6348	Red

Table 7.17: Test case VI

The outcome of this test case unfolds as follows:

- For Service 1, the calculated value stands at 19.7952 of bandwidth, adding to 16.0402 for CPU, yielding a total of 35.8354.
- Service 2, on the other hand, contributes 16.8884 (bandwidth) multiplied by 2 (links used), accompanied by 9.333 for CPU, summing up to 42.1098.

Node	Memory (Gb)	CPU (Gb)
0	241.2431	39.3334
1	16.8426	14.0135
2	7.3443	6.3426
3	7.3443	6.3426
4	7.3443	6.3426

Table 7.18: Memory and CPU Specifications for Nodes

- Service 3, on the other hand, contributes 2.4179 (bandwidth), accompanied by 13.6389 for CPU, summing up to 16.0568.
- Service 4, on the other hand, contributes 3.0744 (bandwidth), accompanied by 5.6348 for CPU, summing up to 8.7092.
- Giving a total Z of 102.7112.

As for the allocation of resources, services 1, 2, and 3 are allocated in node 0, while service 4 takes residence in node 3.

7.4.2 Intermediate Tests: Growing Complexity and Random Scenarios

Building upon the initial phase, intermediate tests involved more complex scenarios. Larger networks and a higher number of services were introduced. To simulate a variety of conditions, random values were incorporated for service resource demands and network link characteristics. This phase assessed the model's consistency and performance in scenarios of increased diversity and intricacy.

In this phase, which involves more intricate scenarios, the concept of slice isolation comes into play. Here, our objective is to observe how our model responds when we employ a multi-slice network. To achieve this, we adjusted the values of the slice isolation variable, denoted as W. Higher values were used to emphasize greater isolation between slices, while lower values were employed to grant the model more flexibility, allowing for analysis of its behavior under different conditions.

The visual representation of these test cases exhibits subtle distinctions due to the utilization of multiple slices within our network. Nodes are color-coded to correspond to their respective slice numbers. The initial slice is depicted in red, followed by the second slice in blue, and finally the third in green. In instances where a node is associated with two slices, it is denoted by a combination of the corresponding colors.

Adjacent to the graph plot, a table provides a comprehensive overview of the parameters associated with the links featured in the graph. Each row in the table corresponds to a specific slice, with the first row pertaining to the first slice, the second row representing the second slice, and so forth.

Test Case 7

Edges	Parameters
(0,1) (0,2) (1,3) (2,3)	Latency: 1 Band: 1000

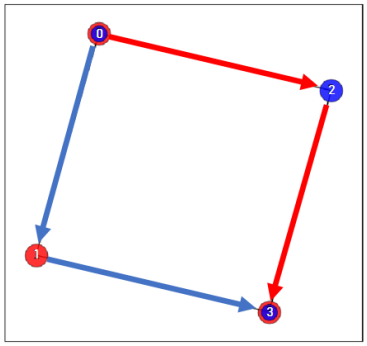


Figure 7.7: Graph Test Case VII

This test case was designed to thoroughly examine the behavior of the penalization parameter. To achieve this, we formulated the following scenario:

- Two services: S1 belonging to slice 1 and S2 belonging to slice 2.
- Source = S and destination = T
- Links (0, 1), (1, 3) are part of slice 1
- Links (0, 2), (2, 3) are part of slice 2
- Node 0 e 3 are shared by both slices
- Node 1 belongs to slice 1 while node 2 belongs to slice 2
- Total available bandwidth for each link: 1000
- Link delay: 1
- Penalization parameter (W): 1000

Service	**Start Node**	**End Node**	**Band.**	**Latency**	**Memory**	**CPU**	**Color**
1	0	3	10	10	10	10	Blue
2	0	3	20	10	10	10	Red

Table 7.19: Test case VII

Node	**Memory (Gb)**	**CPU(Gb)**
0	0	0
1	1000	1000
2	1000	1000
3	0	0

Table 7.20: Memory and CPU Specifications for Nodes

This instance serves to evaluate the accurate implementation of penalization. To thoroughly evaluate this aspect, we deliberately configured W to be equal to 1000, thereby imposing a strong level of isolation between slices. With such a high W value, the model is effectively prevented from choosing links associated with a different service, as doing so would result in an exceedingly high Z value. Consequently, the first service exclusively utilizes the links within its designated slice (red nodes). Similarly, the second service confines its link selection to the links of its corresponding slice (blue nodes). This example effectively validates the functionality of the isolation parameter as intended.

The outcome of this test is as follows:

- Slice 1 = 10 + 10 + 10 = 30
- Slice 2 = 20 + 20 + 10 = 50
- $Z = 50$
- S1 allocates his resources in node 1 and S2 in node 2.

In line with the min-max approach, our goal is to ensure that the worst-case scenario, in terms of fairness, is as good as possible. To do this, we want to maximize the minimum fairness metric among all the different slices in our network.

When we minimize the maximum allocation, we are aiming to minimize the worst-case scenario or the maximum amount assigned to any individual item or entity, which in this case is slice 2 which exhibits a fairness metric of 50.

When considering the slice isolation metric, the model's performance can be deemed commendable. The use of a high isolation parameter, denoted as $W = 1000$, ensured that the model adhered closely to the assigned slices for each service. This is evident from the fairness metric, which achieved a score of 50. Notably, the penalty factor for utilizing a slice not associated with a service was set at 1000. As a result, this value did not contribute to the final outcome. This observation reaffirms that the model successfully upheld slice isolation as anticipated.

Test Case 8

This test case introduces a heightened sense of realism by integrating concrete values for both services and slices. To further illustrate the concept of color-coded nodes, these nodes are categorized into three groups:

- Slice 1 represented in red
- Slice 2 represented in blue
- Slice 3 represented in green

Additionally, it's worth emphasizing the significance of the values enclosed within parentheses. These values denote the normalized bandwidth and CPU values, as elaborated upon in Chapter 6. The outlined scenario encompasses the following set of parameters:

Edges	Parameters
(0,1) (1,2) (0,5) (1,6)	Latency: 10 Band: 978199.5085
(2,3) (3,4) (2,7) (3,8)	Latency: 50 Band: 99724.1929
(8,9)	Lateny: 0.8914 Band: 130817.144

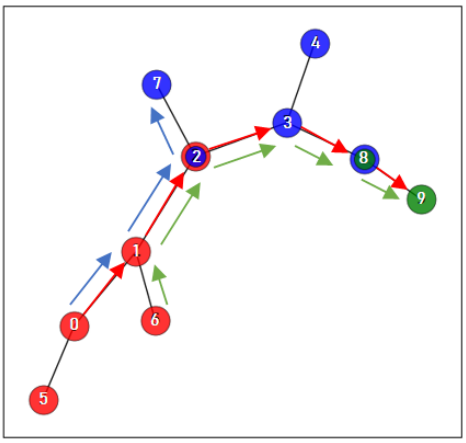


Figure 7.8: Graph Test Case VIII

- 3 services: S1 belonging to slice 1, S2 belonging to slice 2 and S3 belonging to slice 3.
- Links (0, 1), (0, 5), (1,2), (1,6) are part of slice 1
- Links (2, 3), (2, 7), (3,4), (3,8) are part of slice 2
- Links (8, 9) are part of slice 3
- Node 2 is shared by both slices 1 and 2
- Node 8 is shared by both slices 2 and 3
- Nodes 0, 1, 5, and 6 belong to slice 1, nodes 3, 4, and 7 belong to slice 2 while node 9 belongs to slice 3
- Penalization parameter (W): 2

Service	Start Node	End Node	Band	Latency	Memory	CPU	Color
1	0	9	1500 (1.123)	245.2181	2.7632	1.978 (1)	Red
2	0	7	24.4 (1)	74.5357	5.8198	3.7655 (1.713)	Blue
3	6	9	12000 (2)	1906.1377	21.0626	4.4843 (2)	Green

Table 7.21: Test case VIII

In this specific test case, we have fine-tuned the penalization factor to account for the distinctive attributes of our network graph. This graph intricately relies on leveraging different slices to establish connections to designated nodes.

To elucidate how the penalization factor influences our results, we perform a series of calculations. For each link that does not belong to the slice assigned to a service, we apply a penalization by multiplying the bandwidth value by the

Node	Memory (Gb)	CPU (Gb)
0	2.1101	2.1963
1	6.149	3.9262
2	28.1229	4.7201
3	2.1101	2.1963
4	6.149	3.9262
5	6.149	3.9262
6	2.1101	2.1963
7	6.149	3.9262
8	2.1101	2.1963
9	2.1101	2.1963

Table 7.22: Memory and CPU Specifications for Nodes

penalization factor plus 1. This addition of 1 is pivotal due to our equation formulation: in the absence of penalization, we need to ensure a value of at least 1 to prevent multiplication by 0. Subsequently, we incorporate the CPU requirement necessitated by the service's allocation to a particular node.

In this particular test case, as previously mentioned, we intentionally adjusted the penalization factor to 2, in contrast to the previous test case. This decision was made to intentionally ease the strictness of slice isolation, allowing us to evaluate the slice isolation metric. The primary objective of this evaluation was to determine if the model correctly recognizes that it can utilize slices not exclusively dedicated to its service for resource allocation.

The results from this test case provided positive feedback, as the model delivered relatively rapid outcomes. This suggests that the relaxation of slice isolation led the model to focus more on identifying the optimal links for resource allocation, behaving as if there were only one slice among the three services.

As a consequence of this meticulous approach, our model adeptly selects the optimal pathways, yielding the subsequent outcomes:

- Slice 1: 1.123 + 1.123 + 1.123*3 + 1.123*3 + 1.123*3 + 1 = 13.456
- Slice 2: 1*3 + 1*3 + 1 + 1.713 = 8.713
- Slice 3: 2*3 + 2*3 + 2*3 + 2*3 + 2 + 2 = 28
- z = 28

As we mentioned before, we are using the min-max approach in our fairness model. This approach ensures minimizing the discrepancies in fairness among slices, ultimately leading to a better, more balanced distribution of resources.

When we minimize the maximum allocation, we are aiming to minimize the worst-case scenario or the maximum amount assigned to any individual item or entity, which in this case is slice 3 which exhibits a fairness metric of 28. Z here represents a crucial measure of fairness, so the fairness achieved in this case test was 28.

As for the allocation, service 1 allocates its resources in node 1, service 2 in node 7, and service 3 in node 2.

Test Case 9

Edges	Parameters
(0,1) (1,2) (0,5) (1,6)	Latency: 10 Band: 923259.7429
(2,3) (3,4) (2,7) (3,8)	Latency: 50 Band: 91315.2501
(6,9) (8,9)	Latency: 8.2532 Band: 131379.4737

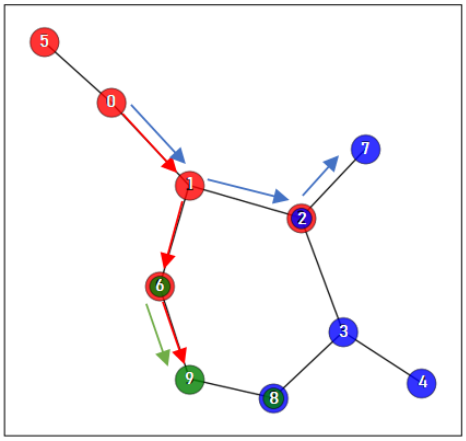


Figure 7.9: Graph Test Case IX

This test case serves as an extension of test case number 8, aiming to explore potential alterations in the model's behavior upon the introduction of a specific link. The inclusion of this link is intended to enhance the provisioning of services, thereby investigating whether the model will opt for utilizing this advantageous connection.

- 3 services: S1 belonging to slice 1, S2 belonging to slice 2 and S3 belonging to slice 3.
- Links (0, 1), (0, 5), (1,2), (1,6) are part of slice 1
- Links (2, 3), (2, 7), (3,4), (3,8) are part of slice 2
- Links (6,9), (8, 9) are part of slice 3
- Node 2 is shared by both slices 1 and 2
- Node 8 is shared by both slices 2 and 3
- Node 6 is shared by both slices 1 and 3
- Nodes 0, 1, and 5 belong to slice 1, nodes 3, 4, and 7 belong to slice 2 and node 9 belongs to slice 3
- Penalization parameter (W): 2

In this test case, we strategically introduced an edge (link(6,9)) to examine how the model's decision-making diverges from its previous behavior, as observed in test case number 8. By introducing a shared node (node 6) between slices 1 and 3, we opened up an alternative pathway for resource allocation to various services.

Service	Start Node	End Node	BW	Latency	Memory	CPU	Color
1	0	9	1500.0 (1.1232)	222.96	2.8382	1.8531 (1)	Red
2	0	7	24.4 (1)	132.07	5.0148	3.7755 (1.6261)	Blue
3	6	9	12000.0 (2)	1758.83	22.790	4.9233 (2)	Green

Table 7.23: Test case IX

Node	Memory (Gb)	CPU (Gb)
0	3.4974	1.9184
1	3.4974	1.9184
2	5.9603	3.8677
3	5.9603	3.8677
4	5.9603	3.8677
5	3.4974	1.9184
6	3.4974	1.9184
7	5.9603	3.8677
8	3.4974	1.9184
9	28.3662	5.0343

Table 7.24: Memory and CPU Specifications for Nodes

Additionally, we adjusted the penalization factor to evaluate the model's behavior. This adjustment revealed that the model now prioritizes resource allocation based on slice utilization.

This test case stands out from the previous one, where the model had only one viable path for resource allocation. In this instance, we provided the model with an alternative option. For example, instead of allocating service 3 to slice 2 as it did in the previous test case, the model, even with relaxed slice isolation constraints, focused on using a link within its own slice to provide service 3. This test case clearly demonstrates that the model comprehends the metrics and their implications accurately.

This shift in the model's behavior offers valuable insights into how its preferences may evolve when a strategically positioned link, such as link(6,9), becomes available.

- slice 1: 1.123 + 1.123 + 1.123*3 + 1 = 6.615
- slice 2: 1*3 + 1*3 + 1 + 1.6261 = 8.6261
- slice 3: 2 + 2 = 4
- z = 8.6261

As for the allocation of the nodes, service 1 is allocated in node 0, service 2 in node 2, and service 3 in node 9.

7.4.3 Real-Scenario Tests: Application in Practical Environments

In the final phase, the model was subjected to real-scenario tests involving practical, multi-slice network environments. These tests evaluated the model's capability to handle real-world complexities, resource demands, and network dynamics. The model's outputs were compared against expected real-world outcomes, further validating its reliability in actual operational scenarios.

Network Slicing for Critical Communications

Edges	Parameters
(0,3) (1,5)	Latency: 300 BW: 5000
(2,3) (3,4) (3,8) (4,5) (5,6) (5,8) (8,11) (8,10) (8,12)	Latency: 10 BW: 10000
(3,7) (5,9)	Latency: 50 BW: 50000

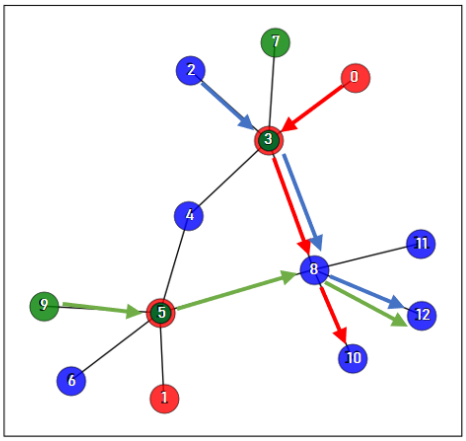


Figure 7.10: Graph Test Case X

- 3 services: S1 belonging to slice 1, S2 belonging to slice 2 and S3 belonging to slice 3.
- Node 3 and 5 are shared by all the slices
- Nodes 0 and 1 belong to slice 1, nodes 2, 4, 6, 8, 10, 11, and 12 belong to slice 2, and nodes 7 and 9 belong to slice 3.
- Penalization parameter (W): 20

Service	Start Node	End Node	BW (kbps)	Latency (ms)	Memory (MB)	CPU (MHz)	Color
1	0	10	500 (1.33)	5000	256	10 (1)	Red
2	2	12	100 (1)	2000	256	10 (1)	Blue
3	9	12	1300 (2)	60000	256	200 (2)	Green

Table 7.25: Test case X

During this test, our aim was to simulate network slicing for critical communications within shared 5G infrastructures. As previously highlighted, this scenario comprises 13 identical servers or nodes. Our objective was to replicate communication patterns within this network, envisioning a situation where a node within a specific slice communicates with a controller located outside that slice. The outcome of this simulation is depicted in Figure 7.10.

Node	0 - 13
Memory (MB)	16000
CPU (MHz)	2200

Table 7.26: Memory and CPU Specifications for Nodes

In this context, the first service initiates its path from node 0, representing the server of slice 1 (Smart Grid), and traverses the network to reach the controller at node 10. Similarly, service 2 commences from node 2 and terminates its communication at node 12, where its corresponding controller is situated. The same pattern is observed for service 3. Notably, our model adeptly selects and utilizes the network links, resulting in improved operational efficiency across all three services.

The outcomes of this test are as follows:

- Slice 1: 1.33 (bandwidth) + 1.33 (bandwidth) * 21 (weight) + 1.33 (bandwidth) * 21 (weight) + 1 (CPU) = 58.19
- Slice 2: 1 + 1 + 1 + 1 = 4
- Slice 3: 2 (bandwidth) + 2 (bandwidth) * 21 (weight) + 2 (bandwidth) * 21 (weight) + 2 (CPU) = 88
- Z: 88

In summary, the overall objective value, denoted as Z, equals 88. This value represents the expected outcome when the model employs the most optimal links to minimize Z. Despite the necessity for the model to utilize slices unrelated to certain services, the meticulous selection of which links to utilize minimizes the influence of the weight parameter on the final result.

Regarding the allocation, Service 1 is assigned to Node 0, Service 2 to Node 2, and Service 3 to Node 5.

SALEM

- 3 services: S1 belonging to slice 1, S2 belonging to slice 2 and S3 belonging to slice 3.
- Node 3 is shared by both slices 2 and 3
- Node 2 is shared by both slices 1 and 2
- Nodes 4,5,6,7,8,9 belong to slice 1, nodes 10,11,12,13,14,15 belong to slice 2 and nodes 0 and 1 belong to slice 3.
- Penalization parameter (W): 20

Due to the constrained time available, we needed to modify our original test plan. The extensive complexity of the network topology, comprising 44 nodes, would

Edges	Parameters
(0,1) (0,2) (0,3) (1,3) (1,2)	Latency: 50 BW: 50000
(2,5) (2,9) (2,4) (2,6) (2,7) (2,8) (4,5) (5,6) (6,7) (7,8) (8,9)	Latency: 300 BW: 5000
(3,10) (3,11) (3,12) (3,13) (3,14) (3,15) (10,11) (11,12) (12,13) (13,14) (14,15)	Latency: 10 BW: 10000

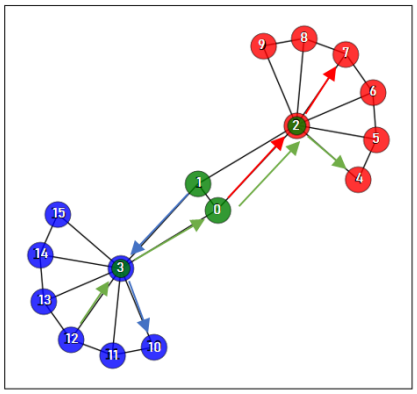


Figure 7.11: Graph Test Case XI

Service	Start Node	End Node	BW (kbps)	Latency (ms)	Memory (MB)	CPU (MIPS)	Color
1	0	15	1500.0 (1.123)	5000	256	20 (1.5)	Red
2	1	15	24.4 (1)	30000	256	30 (2)	Blue
3	2	15	12000.0 (2)	5000	256	10 (1)	Green

Table 7.27: Test case XI

have demanded weeks to comprehensively analyze all possible scenarios. To address this challenge, we streamlined the test to include 16 nodes while still yielding comparable results.

In this adapted test, our focus shifted to simulating switch and cluster configurations. We devised a network structure with two centralized switches, assigned to control network slices within slice 3. These switches were each connected to two cluster switches, fostering communication between devices and the respective slices. One cluster switch was designated for slice 1, while the other was designated for slice 2. Each cluster incorporated six edge switches to establish connections with end devices.

In terms of resource allocation, we determined that service 1 would be allocated to node 2, service 2 to node 3, and service 3 to node 0. The resulting resource distribution can be summarized as follows:

- Slice 1: 1.123 * 21 + 1.123 + 1.5 = 47.166
- Slice 2: 1 * 21 + 1 + 2 = 24
- Slice 3: 2 * 21 + 2 + 2 + 2 * 21 + 1 = 89

The total objective value, represented as 'z,' stands at 89. Our primary aim revolves around verifying the correctness of the solution concerning the problem's requirements and objectives. The achieved result aligns with our expectations, as

Node	Memory (MB)	CPU (MIPS)
0	8000	16000
1	4000	9600
2	8000	16000
3	8000	16000
4	4000	9600
5	4000	9600
6	8000	16000
7	8000	16000
8	8000	16000
9	4000	9600
10	4000	9600
11	4000	9600
12	8000	16000
13	131072	30000000
14	8000	16000
15	4000	9600

Table 7.28: Memory and CPU Specifications for Nodes

the model utilized the available links exclusively to produce the optimal outcome within the imposed conditions.

OREOS Smart City

Edges	Parameters
(0,3) (3,6) (6,9) (9,12) (12, 15)	Latency: 300 BW: 5000
(1,4) (4,7) (10, 13) (6,7) (7,10)(13,15)	Latency: 10 BW: 10000
(2,5) (5,8) (8, 11) (11,14) (14,15) (7,8)	Latency: 50 BW: 50000

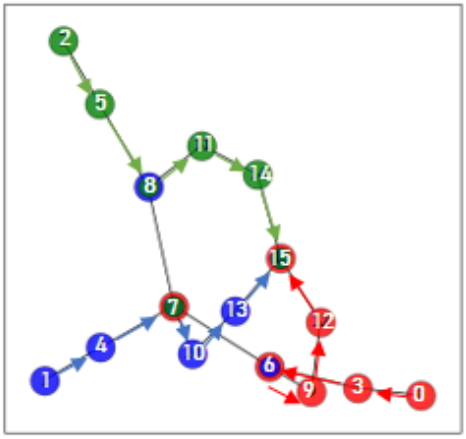


Figure 7.12: Graph Test Case XII

Due to limited computational resources, we refrained from conducting the test with 25 nodes. The processing time for such a scenario could have been significantly prolonged, contingent on the complexity of the network topology. Consequently, we imposed a cap of 16 nodes, enabling us to obtain practical results for evaluation. The scenario was designed as follows:

- 3 services: S1 belonging to slice 1, S2 belonging to slice 2 and S3 belonging

to slice 3.

- Node 6 is shared by both slices 1 and 2
- Node 8 is shared by both slices 2 and 3
- Nodes 7 and 15 are shared by slices 1, 2, and 3
- Nodes 0, 3, 9, and 12 belong to slice 1, nodes 1, 4, 10, and 13 belong to slice 2, and nodes 2, 5, 11, and 14 belong to slice 3.
- Penalization parameter (W): 20

Service	Start Node	End Node	BW (kbps)	Latency (ms)	Memory (MB)	CPU (MIPS)	Color
1	0	15	1500.0 (2)	120000	256	700 (1.87)	Red
2	1	15	500 (1)	5000	256	10 (1)	Blue
3	2	15	1500 (2)	120000	256	800 (2)	Green

Table 7.29: Test case XII

Node	Memory (MB)	CPU (MIPS)
0	8000	16000
1	4000	9600
2	8000	16000
3	8000	16000
4	4000	9600
5	4000	9600
6	8000	16000
7	8000	16000
8	8000	16000
9	4000	9600
10	4000	9600
11	4000	9600
12	8000	16000
13	131072	30000000
14	8000	16000
15	4000	9600

Table 7.30: Memory and CPU Specifications for Nodes

This test case is designed to assess the model's behavior in a real-world scenario by utilizing actual operational values. The objective is to determine if our model is well-prepared for implementation in practical environments. The positive outcome of this test case reinforces our confidence in the model's effectiveness. It is aligned perfectly with the expected results, indicating that the model successfully adhered to the links and slices assigned to each service, consistently selecting the optimal paths.

- slice 1: 2 * 5(links) + 1.87 = 11.87
- slice 2: 1 * 5(links) + 1 = 6
- slice 3: 2 * 5(links) + 2 = 12
- z = 12

As for the allocation service 1 is allocated in node 0, service 2 in node 1, and service 3 in node 2

7.5 Summary

This chapter presents a comprehensive investigation into the optimization of network slicing within the dynamic context of multi-slice 5G networks. The research is structured around a meticulous validation methodology that encompasses analytical validation, diverse scenario testing, and real-scenario validation. These stages collectively ensure the accuracy and practical viability of the proposed model.

The validation process begins with analytical tests that validate the model's outcomes against manually calculated results, ensuring the model's adherence to mathematical principles. Moving forward, intermediate tests introduce complexity by integrating random values for service demands and network attributes. This phase evaluates the model's performance under various conditions, demonstrating its consistency and adaptability.

The highest level of validation is achieved through real-scenario tests, where the model's outputs are aligned with expected outcomes in practical multi-slice network environments. This phase serves as a rigorous assessment of the model's reliability and effectiveness, confirming its applicability in real-world scenarios.

The chapter examines the model's behavior across multiple real-world scenarios. It delves into network slicing for critical communications, showcasing the model's resource allocation capabilities in shared 5G infrastructures. The SALEM scenario simulates a smart city network, evaluating the model's performance in intricate urban infrastructures. The OREOS Smart City scenario tests the model's optimization skills in complex smart city networks.

The datasets and input parameters used in these scenarios encompass service characteristics, node specifications, and the introduction of a penalization parameter. These elements collectively define the model's decision-making process and resource allocation strategies.

In conclusion, this chapter contributes significantly to the optimization of network slicing in multi-slice 5G networks. Through a rigorous validation methodology and diverse real-world scenario exploration, the research provides evidence of the model's efficiency, adaptability, and applicability in practical network environments. The findings underscore the model's potential to enhance resource allocation in the dynamic landscape of 5G networks.

Chapter 8

Conclusions and Next Steps

In the dynamic landscape of modern telecommunications, the dawn of 5G networks illuminates a realm of unprecedented possibilities. This thesis embarked on a journey through the intricacies of network slicing, resource allocation, and the cardinal principle of fairness in network slicing. The culmination of this journey exceeds the boundaries of academia as we stand at the intersection of theoretical investigation and practical application, announcing a significant revolution in network orchestration.

Our work in this thesis embarked on a profound exploration of the potentialities woven within the fabric of 5G networks. The architectural prowess of 5G, underscored by its speed, low latency, and expansive capacity, forms the canvas upon which the concept of network slicing paints its innovation. The opening chapters unveiled the panorama of 5G's promise — a landscape of augmented communication, vehicular autonomy, smart cities, and the sprawling Internet of Things, all framed by the dynamic canvases of network slices.

The Fairness Model, which embodies the need for fair distribution of resources, beats at the center of our discussion. This model emerged from the confluence of meticulous research, mathematical refinement, and the unwavering pursuit of equitable allocation. The model's trajectory wasn't solely towards efficiency; it aspired to uphold the very essence of fairness and equality in the dynamic dance of resource orchestration.

In this comprehensive Network Resource Allocation Framework, with the combining NetworkX, MiniZinc, and Matplotlib we were able to effectively handle simulation, modeling, and visualization of resource allocation. Rigorous testing and validation have transformed this idea into a practical tool, set to reshape resource allocation practices.

The model's strength was put to the test in a variety of challenging contexts, ranging from crucial communications to smart cities. Each scenario demonstrated the model's skill in navigating complex networks and supporting a variety of service and user requests while still adhering to real-world limits.

Yet, as we close this chapter, we find it's not an ending, but rather a transition. The next step of this work is to apply our model within the ONAP framework. The

initial compass, which was focused on making network slicing possible within ONAP, changed into a trajectory that was in line with resource allocation. A framework that was created in conjunction with ONAP's technological environment became apparent as an indication of upcoming implementation. Now that the framework has been verified, we can go on to the last stage, which is integrating it into the ONAP ecosystem.

This thesis goes beyond just being a collection of facts in the weave of invention and application. It marks the beginning of a new era in which resource allocation is wise, network orchestration is efficient, and the symphony of 5G networks reverberates with fairness and equity.

References

What is 5g? differences between 4g and 5g characteristic (embb, urllc, mmtc). URL https://teppeilog.com/whatis5g-e/.

5g vs. 4g: Learn the key differences between them. https://www.techtarget.com/searchnetworking/feature/A-deep-dive-into-the-differences-between-4G-and-5G-networks.

Jgrapht. URL https://jgrapht.org/.

Mosaic 5g on kubernetes with m5g operator. URL https://hackmd.io/@nadhifmr/BkJ3PXXOB.

M-cord. URL https://opennetworking.org/m-cord/.

What is multi-access edge computing?: Juniper networks. URL https://www.juniper.net/us/en/research-topics/what-is-multi-access-edge-computing.htm.

Minizinc. https://www.minizinc.org/.

Mpirical 5g core network. URL https://www.youtube.com/watch?v=YVoCpqsPwmQ&t=484s&ab_channel=Mpirical.

Networkx: Network analysis in python. https://networkx.org/.

Opendaylight. URL https://www.opendaylight.org/.

Onap architecture. URL https://docs.onap.org/projects/onap-optf-osdf/en/latest/sections/architecture.html.

Oreos: Orchestration and resource optimization for reliable and low-latency services. URL https://www.cisuc.uc.pt/en/projects/oreos-orchestration-and-resource-optimization-for-reliable-and-low-latency-ser

Scale-free networks. URL https://www.jsums.edu/nmeghanathan/files/2015/08/CSC-641-Fall2015-Module-5-Scale-free-Networks.pdf.

Kubernetes overview. URL https://kubernetes.io/docs/concepts/overview/.

Oai mosaic5g project group. URL https://openairinterface.org/mosaic5g/.

Open network automation platform. URL https://www.onap.org/.

David Perez Abreu, Karima Velasquez, Luis Paquete, Marilia Curado, and Edmundo Monteiro. Resilient service chains through smart replication. *IEEE Access*, 8:187021–187036, 2020.

Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453, 2018.

Ravindra K Ahuja, Thomas L Magnanti, James B Orlin, and MR Reddy. Applications of network optimization. *Handbooks in Operations Research and Management Science*, 7:1–83, 1995.

Nadine Akkari and Nikos Dimitriou. *Computer Networks*, 169:107082, 2020. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2019.107082. URL `https://www.sciencedirect.com/science/article/pii/S1389128619306346`.

Alcardo Alex Barakabitze, Arslan Ahmad, Rashid Mijumbi, and Andrew Hines. 5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167:106984, 2020.

Luca Cominardi, Fabio Giust, Carlos Bernardos, and Antonio de la Oliva. Distributed mobility management solutions for next mobile network architectures. *Computer Networks*, 121, 04 2017. doi: 10.1016/j.comnet.2017.04.008.

Universidade de Coimbra. Metodologia de avaliação e especificação de casos de uso para validação. *Computer Networks*, v1.0, 01 2021a.

Universidade de Coimbra. Estudos preliminares na área do projeto. *Computer Networks*, 121, 04 2021b. doi: 10.1016/j.comnet.2017.04.008.

SHI Huaizhou, R Venkatesha Prasad, Ertan Onur, and IGMM Niemegeers. Fairness in wireless networks: Issues, measures and challenges. *IEEE Communications Surveys & Tutorials*, 16(1):5–24, 2013.

Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 21, 1984.

Fabian Kurtz, Caner Bektas, Nils Dorsch, and Christian Wietfeld. Network slicing for critical communications in shared 5g infrastructures-an empirical evaluation. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 393–399. IEEE, 2018.

Yuanwen Liu, Bruno Clerckx, and Petar Popovski. Network slicing for embb, urllc, and mmtc: An uplink rate-splitting multiple access approach. *IEEE Transactions on Wireless Communications*, 2023.

Max O Lorenz. Methods of measuring the concentration of wealth. *Publications of the American statistical association*, 9(70):209–219, 1905.

Max O Lorenz. Variabilita e mutabilita. *Publications of the American Statistical Association*, 1912.

Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

A Model. Barabási–albert model.

Matteo Nerini. 5g network slicing for wi-fi networks. `https://github.com/matteonerini/5g-network-slicing-for-wifi-networks.git`, 2020.

Bozidar Radunovic and Jean-Yves Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on networking*, 15(5):1073–1083, 2007.

Bogdan Rusti, Horia Stefanescu, Marius Iordache, Jean Ghenta, Catalin Brezeanu, and Cristian Patachia. Deploying smart city components for 5g network slicing. In *2019 European Conference on Networks and Communications (EuCNC)*, pages 149–154. IEEE, 2019a.

Bogdan Rusti, Horia Stefanescu, Marius Iordache, Jean Ghenta, Cristian Patachia, Panagiotis Gouvas, Anastasios Zafeiropoulos, Eleni Fotopoulou, Qi Wang, and Jose Alcaraz Calero. 5g smart city vertical slice. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 13–19. IEEE, 2019b.

C Shannon and W Weaver. The mathematical theory of communication. urbana: Univ. illinois press. 117 p. 1949.

Henrique Silva, Noé Godinho, and Bruno Sousa. Salem: Service fairness in wireless mesh environments. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2023.

Bruno Miguel Sousa, Henrique M Simões Silva, Noé Godinho, and Marilia Curado. Service function chaining in wildfire scenarios. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 388–394. IEEE, 2021.

Anurag Thantharate, Cory Beard, Rahul Paropkari, and Vijay Walunj. Crawdad umkc/networkslicing5g, 2022. URL `https://dx.doi.org/10.15783/k0w0-js18`.

Zonghua Zhang and Ahmed Meddahi. *Security in network functions virtualization*. Elsevier, 2017.

Appendices

Appendix A

Aditional Information

Service	Service Function (SF)	Description	Comp. cost (MIPS)
Voice PTT	Vo_conf	Conf. voice channels	[1, 30]
	Vo_chan	Voice bridge & channels	[10, 50]
	Vo_core	Voice core, record & log	[10, 50]
Voice FD	VoFD_down	Voice bridge & channels from CCC to FR	[10, 50]
	VoFD_up	Voice bridge & channels from FR to CCC	[10, 50]
Loc. Data	LocData	GPS location data	[1, 10]
	LocData_map	Map location data	[10, 20]
IoT	IoT_acq	Data acquisition from biosensors, sensors	[1, 10]
	IoT_proc	IoT data processing	[10, 20]
	IoT_ana	IoT analysis - risk maps	[20, 40]
Multimedia	Vid_conf	Video conf. (e.g. groups)	[10, 50]
	Vid_ana	Video real-time analysis	[700, 2000]
	Vid_trans	Video transmission	[50, 200]
Security	FW	Firewall & AAA	[20, 100]
	IDPS	IDPS	[50, 150]
Mission Analysis	SF_Mis_ana	Mission analysis	[10, 200]
	Mis_sndData	Upload data to cloud	[20, 40]

Figure A.1: Service Funtions [Service Function Chaining in Wildfire Scenarios]