POLITECNICO DI TORINO

Master's Degree in Communication and Computer Networks Engineering

Master's Degree Thesis

Assessment of an open-source mobile network framework for implementing Network Slicing

Academic and Company Supervisors

Prof. Claudio Ettore CASETTI

Daniele BREVI

Edoardo BONETTO

Candidate

Giulia BERNARDI

2021-2022

Abstract

With the evolution of wireless communication systems, fitting different scenarios with the same physical network infrastructure became infeasible and inefficient, especially with the increasing demand for services and applications. Furthermore, the typical usage scenarios, such as eMBB, URLLC, and mMTC, require more and more divergent needs in terms of bandwidth, latency, and reliability. Hence, to fulfill the market requests, network programmability and virtualization become fundamental. A key enabler for network customization is Network Slicing (NS), which allows overlaying the physical infrastructure with multiple virtual networks and sharing resources efficiently. Therefore, the thesis deals with the inspection of Network Slicing capabilities through an open-source testbed, deploying a small-scale mobile network to simulate a typical situation, where eMBB and URLLC services take place at the same time and share the infrastructure. A cost-efficient way of testing solutions and exploring new ideas is by deploying open-source testbeds with functionalities close to real networks. However, despite the numerous proposals pushed by the industry and academia world, many of them are not easily accessible. Aiming to find the most suitable testing environment, the initial part of the thesis deals with a survey of the principal state-of-the-art open-source frameworks which enable NS. However, among the several solutions, Mosaic5G ecosystem appears to be the most attractive in terms of openness, flexibility, stage of completion, and easy interaction. Therefore, the thesis focuses on Mosaic5G framework capabilities, especially for resource management in RAN segment. Through the testbed deployed, it has been possible to simulate a mobile network, instantiate network slices and manage their life cycle according to dynamic consumer needs. Network performances have been evaluated in different situations, i.e. different numbers of Users Equipment, scheduling algorithms, or channel conditions, analyzing the benefits achieved by introducing slicing in the RAN domain. Experimental results verified the effectiveness of the network slicing approach to satisfy QoS requirements, especially in reducing URLLC latency. Finally, some policies have been deployed to help the automation of the slice management process, aiming to reconfigure the resources on-the-fly and on demand. Mosaic5G resulted to be a powerful tool to quickly deploy small-scale testbeds for prototyping mobile networks supporting RAN slicing, testing use-case scenarios, and developing customized applications.

Table of Contents

List of Tables

List of Figures

Acronyms

IoT
Internet of Things

V2N
Vehicle-to-Network

SP
Service Provider

NS
Network Slicing

EPC
Evolved Packet Core

RAN
Radio Access Network

UE
User Equipment

CN
Core Network

E2E
End-to-End

LTE
Long Term Evolution

OAI

Open Air Interface

QoS

Quality of Service

SDR

Sofrtware-Defined Radio

BTS

Base Transceiver Station

URLLC

Ultra-Reliable Low Latency Communication

eMBB

enhance Mobile Broadband

mMTC

massive Machine Type Communications

CQI

Channel Quality Indicator

RR

Round Robin

PF

Proportional Fair

MT

Maximum Throughput

RTT

Round Trip Time

DP

Data Plane

CP

Control Plane

SDN

Software Defined Network

VIM

Virtualized Infrastructure Manger

VNFM

VNF Manager

NFVO

NFV Orchestrator

CUPS

Control and User Plane Separation

PLMN

Public Land Mobile Network Identifier

MCC

Mobile Country Code

MNC

Mobile Network Code

GUMMEI

Globally Unique MME Identity

TAI

Tracking Area Identity

GUTI

Globally Unique Temporary Identity

nFAPI

Functional Application Platform Interface

PGW

The Packet Gateway

SGW

Service Gateway

MME

Mobility Management Entity

HSS

Home Subscription Server

PDN

Packet Data Networks

E-UTRAN

Evolved UMTS Terrestrial Radio Access Network

PDCP

Packet Data Convergence Protocol

RRC

Radio Resource Control

SO

Slice Orchestrator

RB

Resource Block

RBG

Resource Block Group

RAT

Radio Access Technology

NAS

Non Access Stratum Protocols

RLC

Radio Link Control

MAC

Medium Access Layer

FDD

Frequency Division Duplex

TTI

Transmission Time Interval

Chapter 1

Introduction

The rapid expansion of cellular networks driven by the exponential growth of devices number led to spiraling demand for resources difficult to sustain. New generations of cellular networks arise with the target of supporting denser networks through their enhancements, although, even more frequently the demand exceeds the real capacity, which has physical limits. Furthermore, the increasing number of applications realized to support verticals, such as e-Health, Internet of Things (IoT), Vehicle-to-Network (V2N), or Green Networking, require customized services according to the Quality of Service (QoS) requirements of each scenario.

The purpose of offering a different level of services to the users requires the Service Provider (SP) an on-the-fly programmable platform able to handle the resources dynamically.
One of the key features enabling flexibility and resource management is Network Slicing (NS). It allows the SP to supply customized services, providing the number of resources necessary to satisfy QoS requirements.
Therefore, in such an enhanced and innovative mobile network scenario, arises the necessity of finding the most efficient way of sharing the resources among the different services.
In this context, the aim of the thesis deals with the analysis of the network performances when exploiting an open-source platform supporting Network Slicing management. Therefore, the early stage of the investigation researches for the optimal open-source testbed supporting resource control among the currently available state-of-the-art small-scale projects.

To analyze the effects of resource management in a mobile network, a small-scale testbed has been deployed. The current landscape of open-source software supporting network slicing includes numerous projects but is difficult to use. Aiming to find the best option to build the testing environment for the analysis, the initial part of the thesis investigates the principal open-source frameworks which

enable end-to-end (E2E) NS. Hence, considering a list of indicators to evaluate the characteristics, in Chapter 3 multiple state-of-the-art testbeds have been compared. For its features, its easy deployment, and its maturity, the project preferred resulted to be Mosaic5G, an ecosystem for building agile 4G/5G Service Platforms.
As it will be explained in Chapter 4, Mosaic5G is the result of a community-led consortium that created an open-source project integrating multiple platforms, such as OAI RAN and CN with FlexRAN, Store or ll-MEC.

During the deployment of the testbed, some difficulties have been encountered. Despite the attractive characteristics described in documents and websites, the real stage of completion of the software was immature to allow the development of a complete 5G network supporting NS management E2E. Frameworks enabling slicing to 5G core and RAN domain were not still released at the beginning of the work. Therefore, the target changed according to the available software, deploying an Long Term Evolution (LTE) network supporting NS in RAN domain. Consequently, this work will analyze a prototype of a service-oriented RAN on top of the OpenAirInterface and Mosaic5G platforms that brings programmability and extensibility to the RAN with a range of network applications with the target of intelligent slicing.
Going further, given some problems with hardware, such as the strict CPU requirements and the expensive SDR (Software-defined radio) devices, the testbed was built without a physical BTS (Base Transceiver Station), but leveraging on a Layer 2 simulator.
The major benefit achieved using a simulation environment, besides the cost reduction, is the possibility to test a larger number of users, up to 127, to analyze the network response when densely populated.

Hence, after the deployment and configuration of CN, RAN and UEs through L2-simulator, FlexRAN module has been built as well.
FlexRAN is an Open-source Implementation of a Flexible and Programmable Platform for Software-Defined Radio Access Networks that allows flexible and programmable control of the underlying RAN infrastructure through the introduction of RAN API and virtualized control functions.
The virtualized control functions can be used thanks to a set of mechanisms designed to permit the delegation of control functions, such as schedulers and mobility managers, from the master controller to the base stations at runtime and the reconfiguration of their parameters on-the-fly in an easy way.
Owing to this run-time programmability, FlexRAN platform turns out to be adaptable to the underlying networking conditions and the specific QoS requirements. Furthermore, it allows the application of different isolated slice configurations to handle multiple groups of users.

After making the testbed operative, the testing phase started and all the results have been shown in Chapter 6. Firstly, it has been conducted a survey on the Mosaic5G platform usage, understanding how to exploit the numerous parameters affecting the testbed performances, and different combinations of those are exploited to analyze the response of the network. Such parameters are scheduling algorithms, size of resources associated with each slice, service type, traffic characteristics, and number of users.
In particular, the main idea behind the simulations relies on the need of finding innovative solutions to enhance mobile networks for supporting autonomous vehicles. Thus, a specific use case has been simulated, which includes URLLC (Ultra-Reliable Low Latency Communication) and eMBB (enhanced Mobile Broadband) services that share the bandwidth simultaneously. Especially, the experimental section is driven by the idea of finding a way to prioritize low-latency traffic to reduce the latency at the eNB.

The evaluation begins with a survey on Round Robin scheduler, testing its behavior with different slice occupancy, variable number of UEs, and variable percentage of URLLC devices, showing how and when NS affects the performances most positively.
Going forward the analysis explores the attitude of the other scheduling algorithms implemented by the software, such as Proportional Fair (PF) and Maximum Throughput (MT), examining how the resources are allocated by each of them when the UEs experience different channel conditions.
In particular, it emerges the optimal response of Proportional Fair in an eMBB/URLLC scenario, which prioritize the UEs with the smaller loads for its implementation. Therefore, some further analyses have been computed to better understand PF trends in more realistic situations, such as when URLLC users experience a fluctuating traffic model. The results show a strong improvement to control the undesired peaks in RTT performances.

To conclude, the last step dealt with the implementation of a policy to automatize the process of resource selection on QoS requirements, such as bandwidth, latency, etc. In particular, the policy is realized for a typical URLLC/eMBB use case where mission-critical users download small payloads and rely on an isolated and fixed slice, while eMBBs receive resources on demand. This approach allows efficient exploitation of the channel bandwidth, adapting the service level to the requirements or the status of the network.

With a broader view of the situation, smarter policies tailored to the service needs can be implemented to reduce the waiting time of users or to improve

channel utilization. In a world in which even more devices will be connected and communication rapidly increases, the physical resource limitations become a serious challenge. Furthermore, the numerous use cases that can be designed thanks to the 4G/5G enhancements, require different QoS characteristics. Therefore, it is more and more important to provide customized services sharing dynamically the physical resource through Network Slicing.

Chapter 2

Network Slicing

2.1 Cellular networks: an overview

Cellular networks are high speed and capacity data communication networks for supporting cellular devices. With the increasing expansion of cellular devices, these networks are exploited not only for phone calls, but become the principal way of communication for several verticals, like sensitive business transactions, e-health emergencies, and mission-critical services.
To support the rapid evolution, the complex architectures of mobile networks are transitioning from monolithic deployments, based on dedicated hardware and private firmware and software, to disaggregated networks relying on open source software running on generic SDR or "agnostic" devices. This software-based design represents a relatively recent evolution in the context of 4G networks, while for 5G architectures has been already considered in their early stages.

The main components of last generations cellular network architectures, are the radio access and core network elements. Moreover, to enable dynamicity, programmability, resource sharing and edgefication of a network, new networking principles such as Software-defined Networking (SDN), Network Function Virtualization (NFV),and Multi-access Edge Computing MEC, and Network Slicing have been introduced [1].
The separation among Radio Access Network (RAN) and Core Network (CN) remains unaltered both in 4G and 5G architectures, however the actual implementation and configuration of these components is different. Particularly, they follow the 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) and NR1 specifications for the RAN, and the Evolved Packet Core (EPC) and 5G Core (5GC) for the CN, respectively. Multiple components of LTE EPC that have been traditionally executed on dedicated hardware, recently have transitioned to

software-based deployments. While as fore the 5GC, instead, has been designed according to a service-based approach.

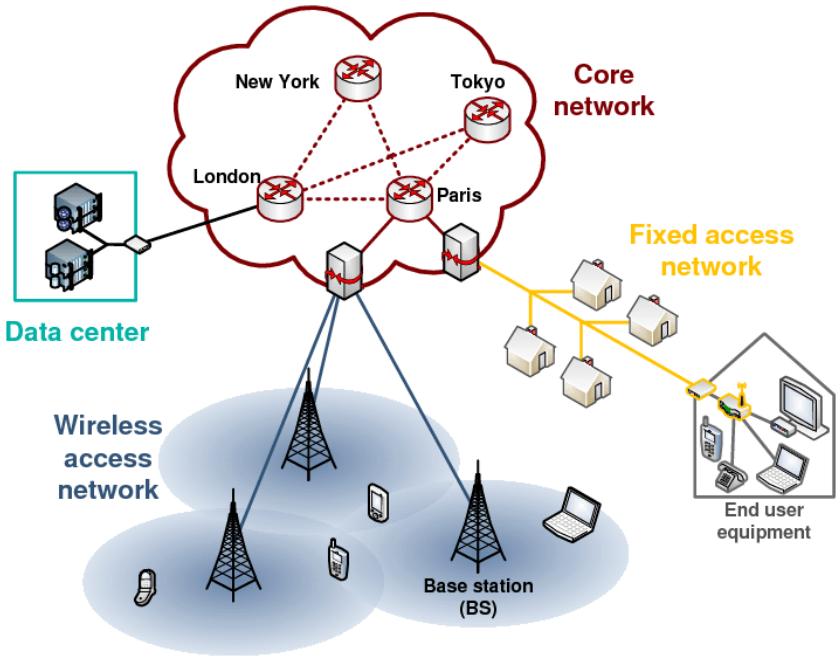

Figure 2.1: Cellular network architecture.

Network Function Virtualization (NFV) and Software Defined Networking (SDN) are the architectural enablers for mobile network virtualization and programmability, and consequently Network Slicing. More in details, SDN facilitates network management through a softwarization approach. It allows the decoupling of Data Plane (DP) from Control Plane (CP), centralizing network management in the SDN controller. SDN controller applications can make use of numerous southbound interfaces (i.e., OpenFlow etc.) to gather network state information and operate on top of each forwarding device. NFV allows the virtualization of Network Functions (VNFs) from hardware components, exploiting general-purpose devices. It enhances the scalability permitting the service provider to distribute new services without introducing new dedicated hardware devices.
Together, SDN and NFV enable dynamic network resource allocation for heterogeneous QoS requirements. In addiction, MEC if used in association to cloud computing, helps to provide reduce latency of services. In parallel to these technologies, it is needed an entity which performs resource management and orchestration efficiently, the so called Management and Network Orchestration (MANO). MANO

is a framework which coordinates physical and virtual networking, storage and compute resources. These resources are required for creating, managing and delivering services through different slices. ETSI has standardized a NFV MANO framework that is composed of three functional blocks connected: Virtualized Infrastructure Manger (VIM), VNF Manager (VNFM) and NFV Orchestrator (NFVO) [2].

2.2 Network Slicing

The huge amount of data necessary to support emerging applications, has resulted in an exponential increasing demand of resources. Such an enormous quantity, it is sustainable only by means of a share utilization of the network, though the division of the physical infrastructure among multiple operators according to their allocation plans. This technique becomes helpful to reduce the Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) costs involved from the operators.

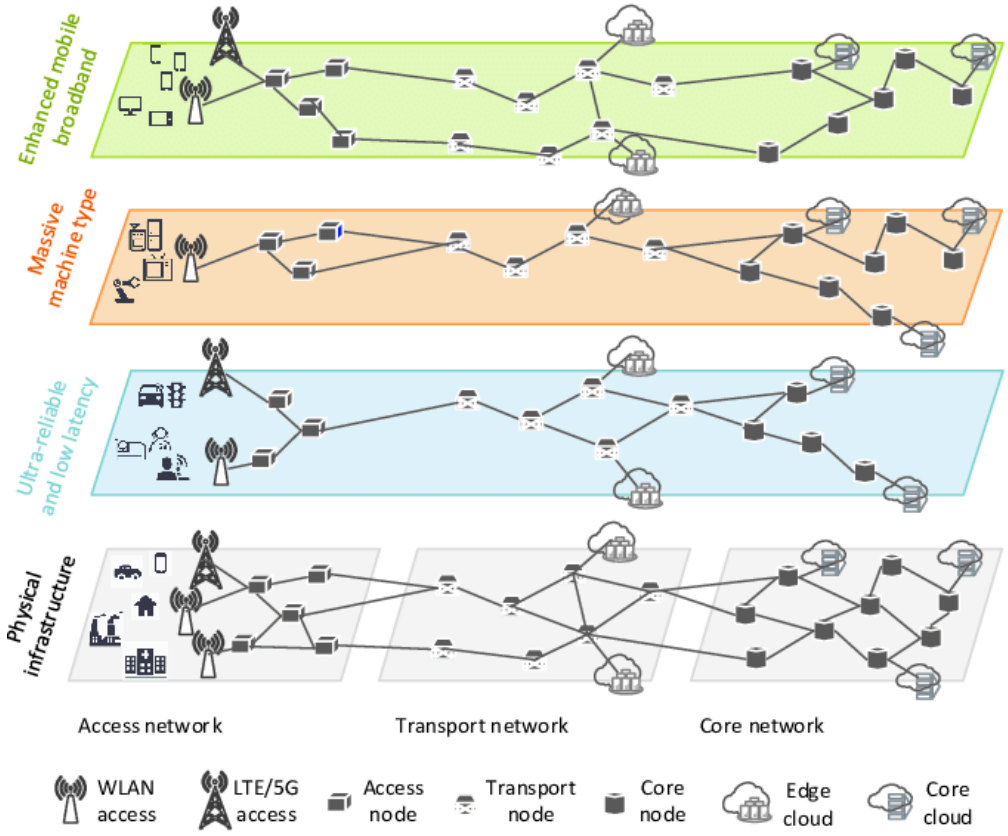

Figure 2.2: Network Slicing over a common physical architecture [3].

The new principles of softwarization, virtualization, and resource sharing, on which new network systems are based, allow the realization of more dynamic and efficient networks. Such a revolution has been made possible, especially by network slicing which is a multi-tenancy virtualization technique where network functionalities are abstracted from hardware and software components, and are provided to the tenants as slices. Therefore, the evolution of network sharing in network slicing brings flexibility and dynamicity to resource allocation. Network slicing allows associating the required amounts of physical resources to different services, over a common physical infrastructure, simultaneously. The physical infrastructure is shared across multiple tenants and each of them can exploit multiple slices.

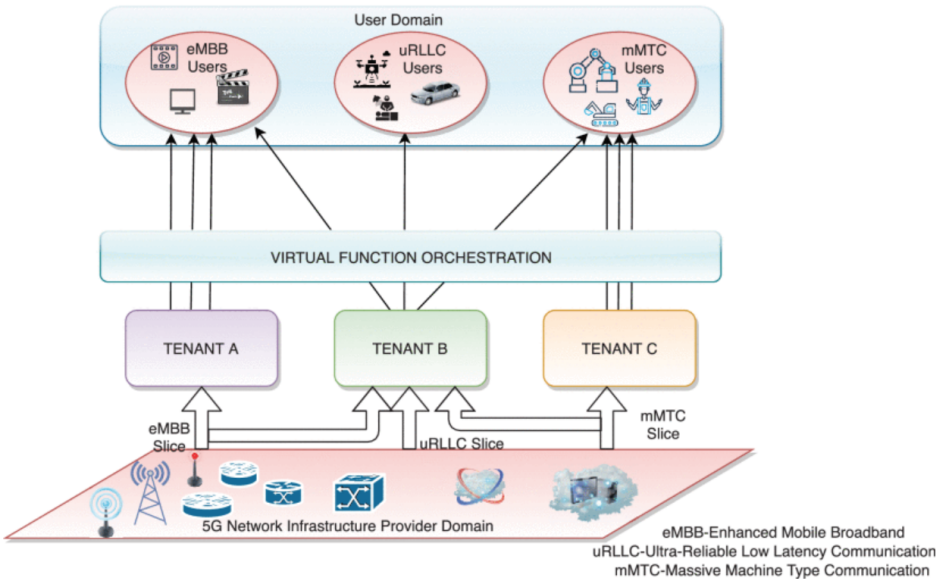

Figure 2.3: NS multi-tenancy architecture [4].

An E2E network slice is a logical, isolated network, generated by NFs chains, which covers all network domains (i.e. RAN, the Transport Network (TN), and Core Network). Each network slice realizes a specific network service according to QoS requirements, leveraging on the underlying physical common infrastructure. And it also provides specific functionalities concerning the RAN and the core part of the network. Each slice is associated with a specific amount of physical resources and operates as an independent virtual network. Moreover, each tenant has complete control of its slices, which are completely isolated from other slices and cannot interact among them. The flexibility of this approach makes it possible to instantiate different slices dedicated to different applications which require sustaining multiple QoS, such as carrying different combinations of resources and priorities at the same time.

Further, Network slicing can operate either on a common shared infrastructure, constituted by generic hardware resources such as Network Function Virtualization Infrastructure (NFVI), or on dedicated hardware (i.e. network elements in the RAN). Focusing on the RAN aspects, the architectural options for RAN slicing cited in the literature [5] are three, where from the left to the right, the slicing depth increases. This implies that the deeper the slicing, the less the network function (NFs) shared among tenants.

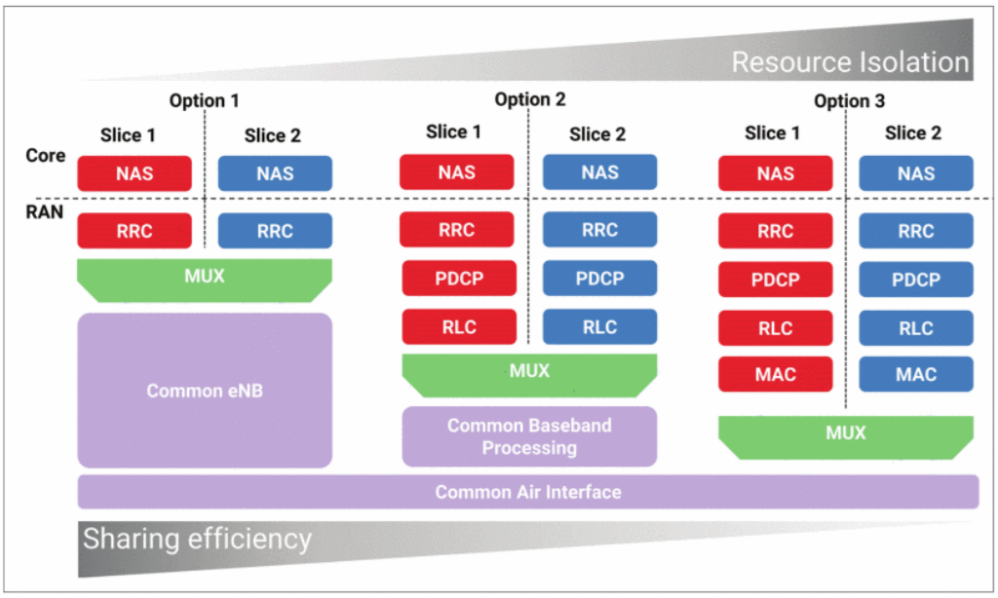

Figure 2.4: RAN architectural options for NS [6].

Starting from option 1, the so called "slice-aware shared RAN", it allows a complete sharing of the RAN, while each tenant is responsible for its CN. The first solution, which can be addressed as the “basic” solution, provides the smaller isolation across tenants, but is the option which offers the highest gains in terms of efficiency.
Moving to option 2, it is found the "slice-specific radio bearer" configuration. In this solution, the depth of slicing is increased along the network stack, and only cell-specific functionality is shared: such as the physical (PHY) and medium access control (MAC) layers in the user plane, and the radio resource control (RRC) in the control plane.
Option 2 extends the resource isolation among tenants, to the disadvantage of the increasing complexity at the MAC layer.
Finally, option 3 addressed as "slice-specific RAN", since all the other functionalities are instantiated specifically for each tenant, while only the air interface is shared among network slices. Thanks to this solution it is possible to provides the maximum flexibility: indeed each network slice can be totally customized, down to the PHY layer. On the other hand, the price to pay for such a freedom, is the tight synchronization requirements between the multi-tenancy policies implemented by a common part and the per-slice (dedicated) implementation [6].

As previously mentioned, the combination of NS in all network domains, such as RAN, TN and CN, allows to an E2E NS system, which is a very important goal to be achieved. Actually, NS has been well discussed in TN and CN domains, but not RAN domain.
Sharing the scarce radio resources in the RAN segment between multiple service providers, while guaranteeing isolation and the customization of functionalities at the same time, it is challenging.
In the recent years, a number of architectures and slice algorithms emerged, however the encapsulation of slices within the RAN supporting variable QoS levels results being still an open problem.

To sum up, the benefits achieved through Network slicing are several. Each slice can be isolated and in charge of handling a specific traffic class, allocating a different amount of resources, as well as necessitating different security requirements. These achievements allow service differentiation at the infrastructure level.
In addition, slicing is controlled by software components, enabling real-time and on-demand instantiation and reconfiguration, adapting the slices to the time-varying traffic demand.
Last but not least, it is possible to exploit the underutilized resources in the form of network slices from the Mobile Virtual Network Operators (MVNOs), in order to maximize resource utilization and reduce the waste [7].

2.2.1 Network Slicing use-cases

With the rapid growth of wireless systems, it has become more evident how different classes of services need to be supported. This is driven by business aspects where third parties, like verticals, require network operators to customize a network on their services.
The increasing diversity of services carried by a mobile network, expands the areas of interest to Industry 4.0, vehicular communications, smart grid, augmented reality, e-Health, and much more.
The complexity of several scenarios is addressed through the standardization of three generic service types, characterized by very different QoS requirements. These, are classified as enhanced Mobile BroadBand (eMBB), massive Machine-Type Communications (mMTC), and Ultra-Reliable and Low-Latency Communications (URLLC). Each of them aims to achieve performance requirements that can improve the user experience of current services and facilitate the deployment of new services.

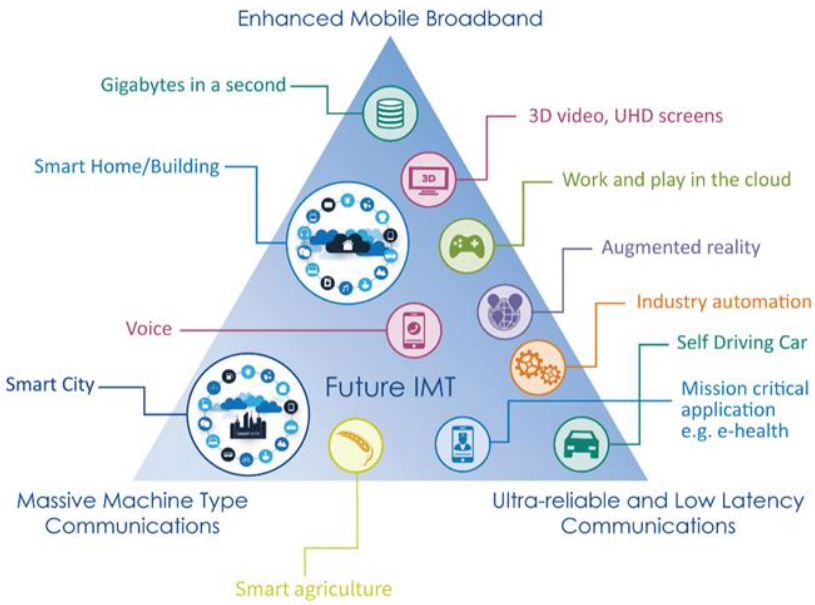

Figure 2.5: Use cases

The three typical 3GPP standardized use-case are characterized as follow:

- eMBB deals with stable connections distinguished by massive data transportation with high peak data rates, without strict latency requirements. It aims to extend the data rates achieved by 4G broadband applications for supporting existing services with better quality or allowing new high-data-rate applications. Services that can take advantage from eMBB are cloud gaming, augmented reality, 3D and ultra high definition (UHD) videos, etc.
- mMTC supports a massive number of Internet of Things (IoT) devices, which are active only occasionally and send small quantities of data.
- URLLC is addressed to support low-latency transmissions of small payloads with strong reliability coming from a limited set of terminals active intermittently, typically depending on outside events. URLLC is considered one of the most relevant enhancements introduced by new generations, indeed the latency of a few milliseconds and the high reliability which should support are expected to be the driving forces behind mission-critical communications, which include services like self-driving cars and industrial automation, reliable remote service or vehicles coordination. URLLC could represent potentially the main enabler of several innovative applications.

Chapter 3

Open-source software for Network Slicing

Slicing has become a key concept in telecommunication systems, since it affects both the business and performance aspects. In this context, the open source community is going through the development of different solutions to allow the integration of slicing algorithms into the 4G and 5G ecosystem.

Besides theoretical aspects for achieving slicing and other networking targets, research communities have followed practical approaches to evaluate the performances of the new concepts in different scenarios. As a result, many prototypes of system implementation of parts of the mobile network architecture have been implemented. These are know as testbeds.
Research testbeds allow to evaluate and enhance the network performances, keeping the deployment costs low and producing a behaviour approximatively comparable with the real networks. Typically, testbeds can be implemented with the help of PCs or servers, not requiring a very high amount of resources and without specialized hardware or software.
Small-scale testbeds are important for research community, since compared with the large-scale ones are more manageable and cheaper, and require less effort to been deployed. Therefore, it is easier to troubleshoot and faster to resolve possible errors.
On the other hand, the practical use cases that can be investigated in small-scale testbeds are limited.

This chapter will survey the most relevant state-of-the-art open source testbed designed for implementing network slicing for RAN and core.

3.1 Indicators: how to choose

In order to design an exhaustive testbed able to emulate a real network, many features should be taken into account [2].

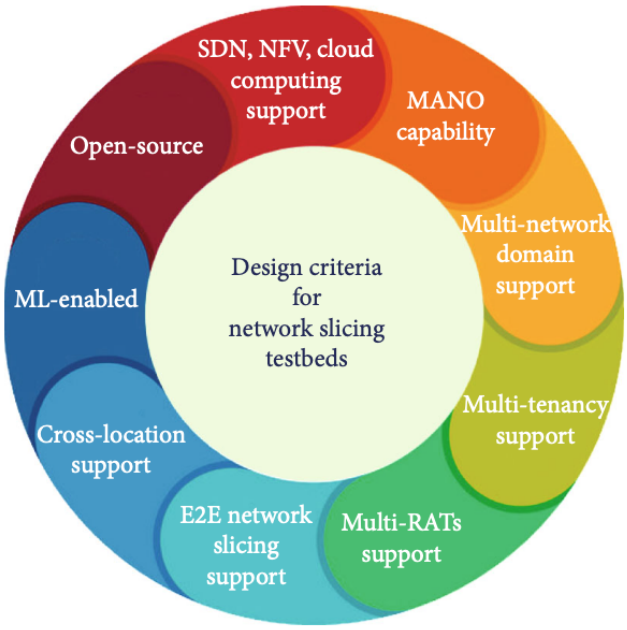

Figure 3.1: Criteria for testbed evaluation [8].

Firstly, the network slicing testbed should support the main enabling technologies, such as SDN, NFV and cloud computing. Thanks to these functionalities, it is possible to grant flexibility and dynamicity in the network.
Furthermore, the testbed should support dynamic monitoring capability: management, programmability and orchestration of many network functions and services. That is possible with the presence of the MANO entity.
Another fundamental feature that a research testbed cannot live without, is the multi-domain support. Indeed, it should provide connectivity around all network domains (RAN, TN, CN) in order to support network slicing E2E.
Nevertheless, network slicing can be implemented even partially, for example in one network domain only.
An important aspect for the last network generations is the capability of multiple tenants of exploiting the same network functionalities, simultaneously. Not simply, but also the ability to interact and manage the cooperation with them. These

competences represent the multi-tenancy environment.
To continue, the testbed should support different Radio Access Technologies (RATs): WiFi, LTE, 5G NR should be deployed over the same platform. Such a feature is called multi-Radio Access Technologies (multi-RAT).
Even if it is not strictly mandatory, the slicing concept should be present in all network domains. In order to perform E2E slice, each slice subnet instances belonging to a different network domain, has to be chain with the others.
In addiction, another feature essential for the testbed is the openness. Open-source environments allow to reduce the complexity of setting up a working mobile network in research world. Thus, an open-source platform is helpful to improve and enhance the studies.
Others additional equipment that can be interesting in the choice of the testbed, could be the enabling of Machine-Learning tools. These can be helpful to predict and verify the channel behaviour in RAN domain when applying network slicing. Accordingly, it would be possible to schedule the radio resources in an optimized way, improving the maximization of the usage per slice [1],[8].

3.2 Open-source testbed: projects description

The prototype of a mobile network with network slicing capabilities requires multiple pieces of software. Quickly summarizing them, it is found: the Radio Access Network (RAN), the Core Network (CN) and the Management and Orchestration (MANO) part.
The major open source solutions to realize mobile networks in software are built on GNU Radio development suite and the Ettus Research USRP SDR platforms, running on Linux-based devices.

As regard for the RAN part, some of the most popular software solutions to run LTE over SDR are OpenAirInterface (OAI), openLTE, and srsLTE.
Despite OAI-RAN experiences better performance in terms of throughput and resource footprint, srsLTE provides a “smaller” source code that makes it easier to customize.
While for the CN, apart from commercial solutions, the most relevant among open-source projects are srsEPC provided by srsLTE, a lightweight CN implementation which is released under the same license. And the CN released by OAI which provides the same elements for an LTE EPC solution (release 10 with a subset of release 14) and can be deployed on standard Linux-based device. OAI-CN is released under a standard Apache v2.0 license.

Testbed	SDN	NFV	Cloud comp.	Multi domain	Multi tenancy	MANO	Open-source	Multi RAT	E2E slicing	ML	MANO
Orion	✓	✓	✓	✓	✓	✓	x	x	x	✓	OSM and customized
Canonical	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	Open Stack
SEMIoTICS	✓	✓	✓	✓	✓	✓	✓	x	✓	x	Open Stack
Mosaic5G	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	JOX
Simula	✓	✓	✓	✓	✓	✓	✓	✓	x	x	OSM
M-CORD	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	XOS
POSENS	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	Customized
CAI	✓	✓	✓	✓	x	x	✓	✓	x	✓	x
5G Testbed for NS	x	✓	✓	✓	x	x	✓	x	✓	x	x
5GIIK	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	OSM
5G Tactile Internet platform	✓	✓	✓	✓	✓	✓	✓	x	✓	x	OpenStack Tacker

Figure 3.2: Comparison of small-scale testbed for network slicing.

Plenty of solutions have been pushed from the research world, although many of them do not include all the design criteria but focus only on single aspects. The figure below (Fig. 3.2) collects some of the most relevant projects, highlighting their main functionalities. Among them, the most interesting solutions for our case (i.e. the one with more features) have been selected. Therefore, the testbeds implementing slicing and leveraging on most of the network principles to enable flexibility and programmability will be described in detail.

3.2.1 Mosaic5G

Mosaic5G [9] is an open-source ecosystem with an infrastructure built on OAI-RAN and OAI-CN, that enables monitoring, control, and programmability of RAN and CN modules through north-bound APIs. Above this infrastructure there are two platforms that provide virtualization and slicing, FlexRAN and LL-MEC. These are the two main controller for RAN, and edge/CN domain. In addition, it introduced an orchestration based on Juju that support network slicing.
The principal frameworks composing the Mosaic5G architecture include FlexRAN [10], which is a programmable platform for Software Defined Radio Access Network,

applying the SDN principles at the RAN domain. FlexRAN, working on OAI-RAN infrastructure, separates the operations between control and the user plane and allows the centralization of the control in RAN domain. The principal enablers of this capabilities are the Real Time Controller (RIC), and the RAN runtime. Then, ll-MEC is an open-source MEC framework for cellular systems 3GPP and ETSI compliant. This framework combines SDN, edge-computing and abstraction principles to provide an end-to-end platform where services are executed at the borders of the network. The principal components are: the Edge Packet Service (EPS) which controls the core elements through OpenFlow APIs, and the Radio Network Information Service that interfaces the data plane with the eNBs through FlexRAN protocol.
Basically, LL-MEC divides the CP and DP traffic at the edge and the CN domain. That way, MEC functionality is realized. To sum up, FlexRAN and LL-MEC combined perform SDN functionality at the RAN, at the edge and at the core domains.

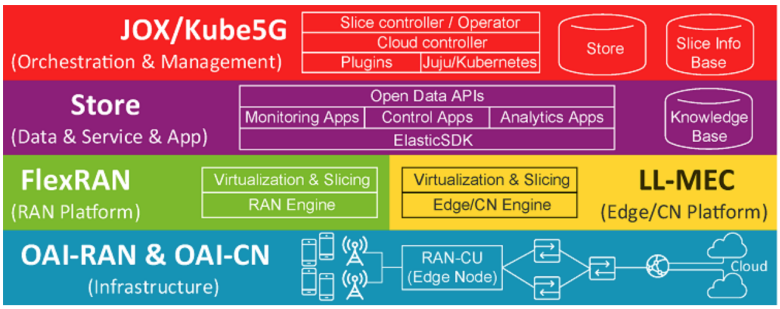

Figure 3.3: Mosaic5G architecture [9].

Furthermore, other frameworks included in Mosaic5G project are OpenStack, Juju, and JOX, that are adopted as VIM, VNFM, and NFVO, respectively. OpenStack is an open source service framework, which provides service reservation and virtualization. Furthermore, OpenStack is supported by a modular and pluggable architecture. Juju, instead, is responsible of installation, configuration and communication among services. However, it does not take the actual decisions for a particular service, but delegates to specific services through a set of scripts named Charms. A Charm defines the ways to install and run a service, as well as how to fill up configuration files or react to events. Further, Juju is exploitable for a quick and efficient deployment, configuration and integration, performing operations in a wide area of public clouds. JOX supports the lifecycle management of a network slice and the mobile network orchestration. Inside the JOX core, a set of services is used to control each network slice, while support the interaction between resource

and service orchestration, VNFM and VIMs simultaneously. [11]

3.2.2 5GIIK

5GIIK [2] is an open source platform which exploits OAI as Core Network, while employs srsLTE for the RAN part. The code of srsLTE is well-structured and easily accessible and its library is modular and to increase its performance in the system, it utilizes single instruction multiple data operations. Moreover, it benefits from a light implementation, and from different RF front-ends interoperability. Both CN and RAN are cloud-based VNFs via OSM deployed on two OpenStack platforms located in different geographical areas, enabling also cross-location. As NFV orchestrator it relies on Open-Source MANO (OSM), developed in python and operating on linux. OSM is one of the best in terms of compatibility with different VIMs and for resource consumption. Moreover OSM employs Docker containers and solutions cloud-based. Lastly, VIM is implemented by OpenStack, which is powerful for infrastructure orchestration, scalability and resource utilization.
Furthermore, two Tenant Controllers (TC) for the whole network domains have been integrated in order to make the design more generic. Hence, 5GIIK testbed uses 5GEmPOWER as TC for RAN domain, which is RAT agnostic, thus can manage the virtualized network resources of multiple radio nodes. And M-CORD as TC for Transport Network and Core Network domains. M-CORD is a cloud based solution built on SDN and NFV technologies that allows virtualization of RAN functions (vRAN) and a virtualized CN (vEPC). WIth the integration of SDN controllers (ONOS) in M-CORD architecture, it is enabled network slicing E2E.
This testbed provides E2E network slicing by defining specific descriptors in a hierarchical way at the VNF, network slice levels on CN and RAN domains. Therefore, 5GIIK is a testbed architecture that grants an E2E network slicing with MANO capability, supports multi-tenancy and multi-RATs, being at the same time a cost-efficient design.

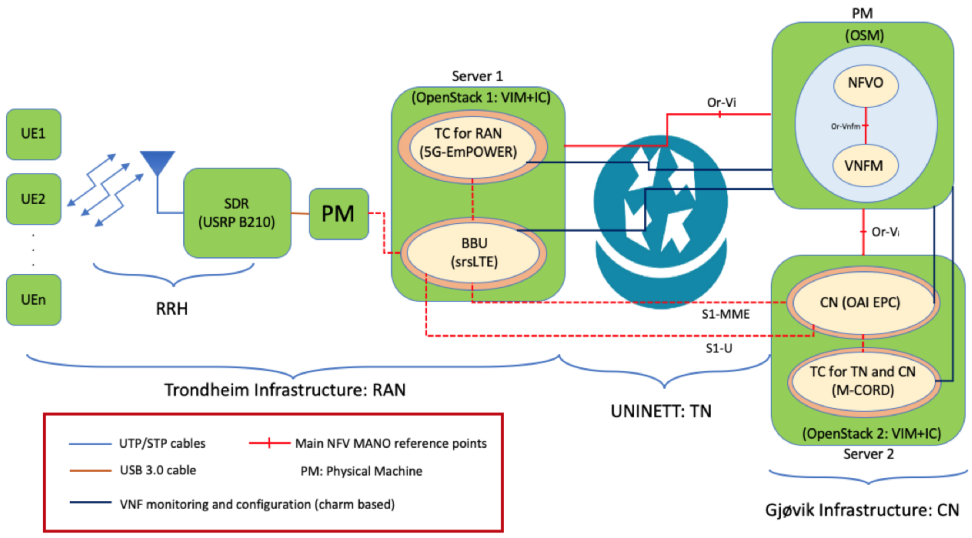

Figure 3.4: 5GIIK architecture [2].

3.2.3 POSENS

POSENS [6], [12] is a platform which provides efficient resource management for the creation of E2E independent and customizable slices. It is an open-source solution that allows the creation of E2E network slices, deploying the testbed with commodity hardware and SDR boards.
As it was shown in Chapter 2, there are three RAN architectural options for implementing NS, each of them with its pros and cons. POSENS in its first release, has decided to implement the first option, realizing a "slice-aware shared RAN" solution, which aims to efficiently sharing the network resources between different tenants.
In particular, POSENS testbed has been deployed with srsLTE as RAN part, an open source implementation of a UE and the eNB, and OAI as CN without any modification. A positive characteristic of POSENS is its interoperability: the solution is able to work with any EPC implementation supporting the S1AP protocol.
While for the MANO part, POSENS wants to provide per-slice management, difficult to find in non-commercial world, Therefore, the project deployed its own orchestration mechanism, exploiting a dedicated software that directly leverages on the VIM APIs, called POSENS MANO. Moreover MANO chains different NFs in the network layer to create network slices.

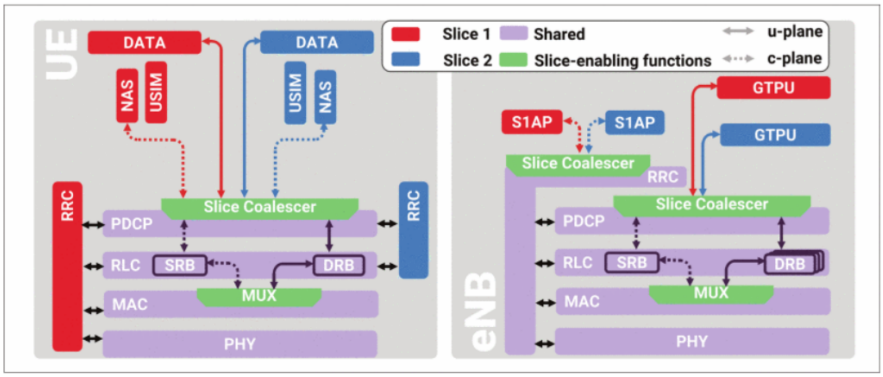

Figure 3.5: Design of POSENS architecture [6].

Therefore, going deeper in POSENS implementation (Fig.3.5), it will be briefly shown the architectural changes introduced to deploy the "slice-aware shared RAN" solution, where slices are multiplexed and demultiplexed at the PDCP layer. This option implies fewer changes in the eNB software implementation, which is one the principal cause of instabilities in an SDR-based testbed. Furthermore, each slice has its own RRC module at the UE, without requiring additional functionalities inside the CN. On the contrary, at the eNB there is only one RRC module, able of managing multiple non-access stratum (NAS) from different users simultaneously. Lastly, the main feature of the solution are the "slice coalescer" modules which can be found at the PDCP layer and above. These modules are responsible for forwarding the control and data layer information for each slice over a common shared channel.

3.2.4 M-CORD

This work [11], [13] focuses on the integration between OAI with the M-CORD platform to deploy LTE network on top of Mobile Central Office Re-architected as a Datacenter (M-CORD).

M-CORD is another option to manage the RAN and CORE softwarization of 5G infrastructure, an open solution for 5G networks provided by the open networking lab (ON-Lab). Several entities are integrated into M-CORD platform, which emulate a complete network. For example, XOS performs service orchestration and OpenStack provides the infrastructure for deploying the services via chaining VNFs. The SDN controller is implemented by Open Network Operating System (ONOS), that separates CP and DP functionalities. ONOS, via its Southbound Interface (SBI), is involved in management procedures on TN. M-CORD also provides an

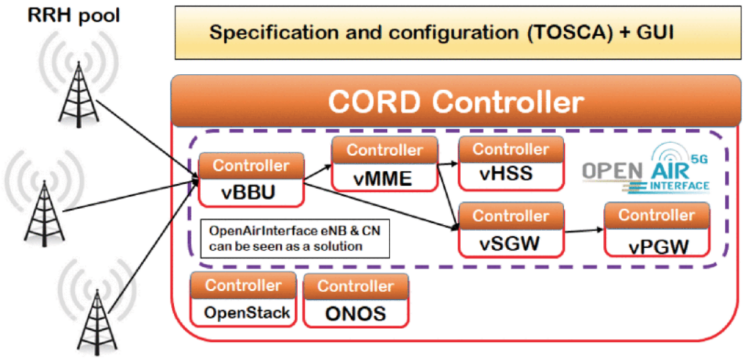

Figure 3.6: M-CORD architecture [11].

interface (GUI) to enabling easy development, modifications and configuration of the resources.
In particular, [13] proposes a framework which exploits M-CORD architecture by introducing a slicing mechanism for transport network (TN) between access and core network. This is possible through an application on top of ONOS which creates dedicated slices and set up various flows between the access and core network depending on the service type.

3.2.5 CANONICAL

The next testbed which is presented, is CANONICAL, a solution proposed by Ubuntu.

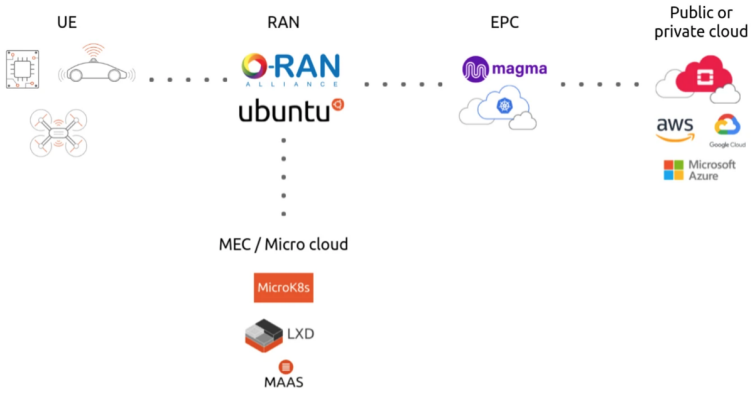

Figure 3.7: CANONICAL architecture.

As for the RAN part, CANONICAL relies on OpenRAN, which is a software for disaggregated and open radio access network, running on an Ubuntu Operating system. It provides open interfaces between components, which allows general purpose HW and SW. The solutions can be implemented on Bare Metal, VM or Container.
The Core Network part, instead, is based on Magma, an open and flexible solution, which provides a state-of-the-art EPC, with Federation Gateway and Access Gateway. Magma provides to service operators an cost-effective and simple infrastructure to build enhanced mobile networks, and it is designed to be 3GPP generation and RAN agnostic. However, at the time of writing the 5GC resulted still under development.
As regard for the community, this software can relies on an active support.
Moreover, it employs Charmed Kubernetes to provide container infrastructure for RAN and EPC and Charmed OpenStack to facilitate the deployment. MAAS LXD and MicroK8s merged together, build a micro cloud which allows to run edge applications as VMs, containers, bare metal and providing all Enhanced Platform Awareness (EPA) features.

3.2.6 5G Tactile Internet platform (SEMIoTICS)

5G Tactile Internet platform [14] employed the SEMIoTICS architecture to build a 5G NFV-enabled experimental platform. That consists of open-source software, which leverages on SEMIoTICS framework extending its capabilities with NFV, SDN, and MEC technologies.

The SEMIoTICS architecture leverages NFV/SDN in a three-layer framework, which consists of field, network, and backend/cloud. Each layer is composed of devices with different characteristics compute, storage, and network characteristics. The aim of SEMIoTICS is to create a 5G platform for providing secure and reliable E2E services to support industrial IoT applications with a very strict latency. The Network layer is responsible for management of the testbed virtual domain and it chains VNFs by utilizing the SDN controller Neutron. Finally, the lower layer, Field layer, establishes connections between sensors and actuators with the upper layers. The communication process takes place through the exchange of messages between virtual SDN switches in the Networking layer and IoT/IIoT gateways in Field layer. The testbed performance has been assessed with the goal of performing E2E slicing and dynamic sharing the bandwidth between two VNFs, one for monitoring purposes and the other for actuating, which are the currently implemented.

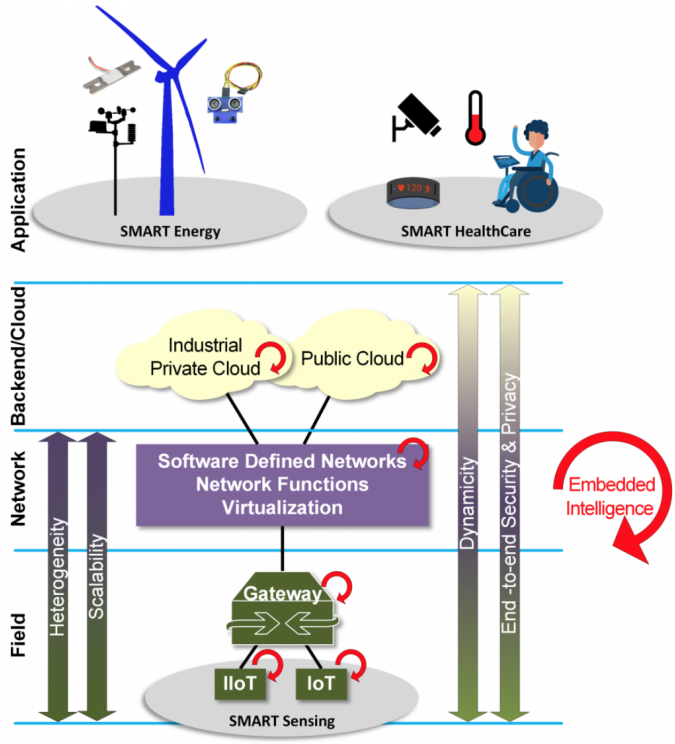

Figure 3.8: SEMIoTICS architecture: framework layers [15].

3.3 Conclusions

To conclude the survey about the more attractive open-source software for NS, many proposals appear interesting and suitable for research purposes to the same extent.
Therefore, since the potentials of each framework concerning the technical features are more or less equivalent, other factors are taken into account. So, for instance, how to deploy or interact with the platforms. Among the open-source projects previously described, Mosaic5G turned out to be the clearest. From a documentation point of view, Mosaic5G seemed to be the more mature and the easier in terms of understanding. Moreover, the platform benefits from a complete integration between the frameworks, and thanks to the snap-based versions of the software, is quite easy to build as well.
Mosaic5G is supported by a large and active community, offering rapid and detailed answers for users and developers, and the website provides detailed documentation, as well as multiple tutorials for deploying the framework or exploiting the capabilities. In addition, some videos about experiments and explanations are available on

the web. For these reasons Mosaic5G is the testbed preferred.

Overall, the gap between theory and practice for 5G networking is still large. Even if Network Slicing has been acknowledged as the enabling technology to support different QoS requirements, the documentation about practical experiments on the use of this technology is still really poor.
Therefore, despite the good prospects perceived from the literature on Mosaic5G, the real state of progress of the platform at the beginning of the deployment phase was different. Most of the frameworks enabling NS in all network domains for a 5G mobile system were in delay.

Chapter 4

Overview of Mosaic5G architecture

Mosaic5G is an ecosystem of open-source platforms for 4G/5G system based on SDN, NFV and MEC. Mosaic5G is build on top of OpenAirInterface (OAI), which is an open-source, software-based and standard-compliant mobile network ecosystem for prototyping LTE and 5G Mobile Networks. The framework Mosaic5G can be used as an open-source lightweight service delivery platform to deploy easily a network testbed and explore new idea of applications.

The initial objective of the work dealt with the deployment of an E2E slicing 5G network, focusing on the analysis of slicing impact on user experience. Unfortunately, as mentioned, despite the features promised by documentation, during the implementation phase it has faced the problem of delayed implementation. Indeed, the software to realize a 5G mobile network with NS covering all network domains was still unavailable. In particular, it was available only the framework for RAN slicing. For this reason, the testbed will be deployed for an LTE system, implementing NS only in RAN domain.

In order to have a clear picture of the implementation, it will be briefly introduced the LTE architecture describing the most important elements of 4G cellular network.

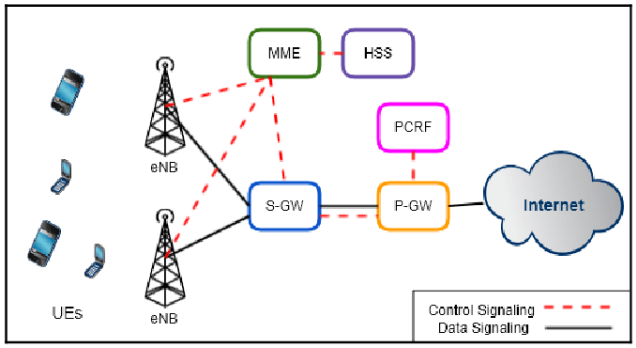

Figure 4.1: LTE Cellular network.

As regard for EPC, the main components are: The Packet Gateway (PGW) and Service Gateway (SGW), which are packet gateways to and from the Internet; the Mobility Management Entity (MME), which handles handovers and the UE connection life cycle from the core network point of view, and the Home Subscription Server (HSS), which manages subscriptions and billing.
The EPC is divided into several virtual network functions, each of them providing specific functionalities, and connected to each other through standardized and open interfaces.

Deeper in EPC elements functionalities, it states that MME is the control-node for the LTE access-network. It is responsible for control messages which are needed to establish a connection with UEs, paging, mobility and tagging procedure, including retransmissions.
It deals with bearer management and it decides the SGW for a UE at the initial attach or at intra-LTE handover. Furthermore, it takes care of users authenticating interacting with the HSS.
It includes the signaling and security features of Non Access Stratum (NAS) and it handles the generation and allocation of temporary UEs identities as well.
MME checks the authorization of the UE to make use of the service provider's Public Land Mobile Network (PLMN) and enforces UE roaming restrictions. The MME also provides the control plane function, supporting the related protocols, for mobility between LTE and 2G/3G access networks.
The main function of the SGW, is routing and forwarding data packets and support the mobility for the user plane among eNB or other 3GPP technologies during handovers, as well. For idle state User Equipment, the Serving Gateway terminates the DL transmissions and triggers paging when data arrives for the UE. Moreover, it stores and manages UE information, and it replicates the user traffic in case of legitimate interception.

The PGW provides connectivity from the UE to external packet data networks (PDNs). The same UE can have simultaneous connectivity with more than one PDN Gateway in order to access multiple packet data networks.
The PDN Gateway is responsible for packet filtering and screening, policy enforcement, charging support, lawful interception. Further, PGW acts as anchor for handover between 3GPP and non-3GPP technologies.

The evolved Node Bases (eNBs), the base stations for LTE, is the main components of LTE RAN, and it provides wireless connectivity to the mobile User Equipments (UEs). The eNBs are typically implemented on dedicated hardware, and are connected together and to core.

LTE eNB Stack is the key software element of 4G base stations, and its radio protocol architecture, shown from Fig. 4.2 and 4.3 (from 3GPP TS 36.300 V10.12.0), is given for the User and the Control plane.

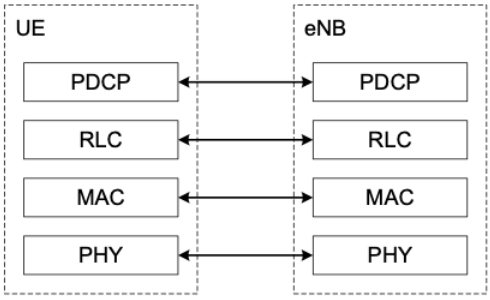

Figure 4.2: RAN protocols UP.

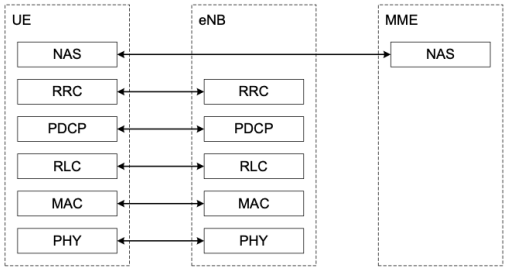

Figure 4.3: RAN protocols CP.

Traditionally, the LTE RAN protocol stack for the UP includes: Packet Data Convergence Control (PDCP), Radio Link Control (RLC) and Medium Access Layer (MAC). The main functionalities of PDCP are header compression, ciphering and integrity protection. RLC operates in 3 modes: Transparent Mode (TM), Unacknowledged Mode (UM), and Acknowledged Mode (AM). RLC Layer is responsible for error correction through ARQ transfer of upper layer PDUs, concatenation, segmentation and reassembly (functions performed only by AM and UM). The TM transfers data transparently. Finally, MAC protocol deals with resource allocation, therefore scheduling and allocating resources to multiple logical channels. While the LTE RAN protocol stack for CP includes also Radio Resource Control (RRC) and Non Access Stratum (NAS) Protocols. RRC is responsible for broadcast System Information related to the NAS and AS (Access Stratum), paging, and for establish, maintain and release an RRC connection between the UE and E-UTRAN. It occupies also security for functions. NAS supports the mobility of the UE, as well as the management procedures to establish and maintain IP

connectivity between the UE and a PDN GW.

4.1 OpenAirInterface

OpenAirInterface (OAI) is an open source project developed by Eurecom, that provides 3GPP standard compliant implementations of the main components of 4G and 5G Radio Access and Core Network. Those software run on Software Defined Radio (SDR) card like the USRP (ETTUS Universal Software Radio Peripheral). To sum up, OAI framework allows to establish a flexible, interoperable and standard compliant 4G/5G network .

The main difference of OAI from other similar projects is its unique open-source license, OAI public license v1.1, which was created by the OAI Software Alliance (OSA) in 2017. It is a reinterpretation of Apache v2.0 License with additional clauses, which allows the free utilization of OAI code for non-commercial and academic research purposes.
This distribution is exploited in order to permit owners of significant patents to contribute to the OAI source code, while keeping their licences rights.
The highlights of OAI Public Licence are the capabilities of allowing contributions from 3GPP member companies and permitting at the same time the commercial exploitation of the code, which is not at all possible with other open-source projects. [16]

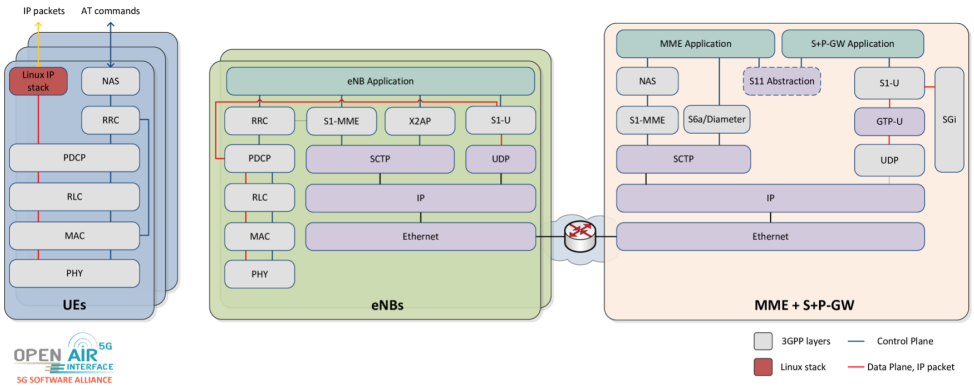

Figure 4.4: OpenAirInterface Block diagram [17].

The OpenAirInterface provides software-based implementations of LTE base stations (eNBs), UEs and the EPC, compliant with LTE Release 10 standard.

4.1.1 Description of OAI-RAN features

Hence, OAI-RAN is the software branch which implements eNBs. The source code is written in C in order to guarantee real-time performance. Moreover, both the eNB and UE implentations are compatible with Intelx86 architectures running the Ubuntu Linux operating system.
As regard for the physical layer implementation of the LTE RAN, it can operate in Frequency Division Duplexing (FDD) and Time Division Duplexing (TDD) with 5, 10 and 20 MHz channel bandwidths, which correspond to 25, 50, 100 Physical Resource Blocks (PRBs), respectively.
The transmission modes supported are Single Input Single Output (SISO), transmit diversity, closed-loop spatial multiplexing, Multi-user Multiple Input, Multiple Output (MIMO and MU-MIMO), and 2 $\times$ 2 MIMO.
Moreover, Channel quality information are reported through standardized Channel Quality Informations (CQIs) and Precoding Matrix Indicatorss (PMIs). Finally, OAI-RAN also supports HARQ at the MAC layer.

In DL the channels supported are PSS, SSS, PBCH, PCFICH, PHICH, PDCCH, PDSCH, PMCH, MPDCCH.
In details, OAI-RAN implements synchronization signals exploited by UEs to collect symbol and frequency synchronization (Primary and Secondary Synchronization Signal (PSS),(SSS)), and channels that carry information on the DL configuration used by the eNB (Physical Broadcast Channel (PBCH)) and on the DL control channel (Physical Control Format Indicator Channel (PCFICH)).
Moreover, OAI-RAN eNB implements as well, the Physical Downlink Control Channel (PDCCH), carrying scheduling assignments of the UEs and DL control information, and the Physical Downlink Shared Channel (PDSCH), which delivers data for specific UEs. Lastly, ACKs/NACKs for the data received in UL from the UEs are sent through the Physical Hybrid ARQ Indicator Channel (PHICH), while broadcast and multicast services are provided through the Physical Multicast Channel (PMCH).

Further, the UL channels supported are: PRACH, PUSCH, PUCCH, SRS and DRS.
The Physical Random Access Channel (PRACH), which handles UEs requests to UL allocation to the base station, and channels carrying reference signals between UE and eNB. Data from the UEs to the eNB is carried by the Physical Uplink Shared Channel (PUSCH), while the Physical Uplink Control Channel (PUCCH) is used to transmit UL control information. [17].

The LTE RAN stack implements the MAC, RLC, PDCP, and RRC layers

and provides interfaces to the Core Network with support for IPv4 and IPv6 connectivity. It is possible to find a complete documentation on all the features supported by each protocol layer in this reference [17].
As for the MAC layer scheduling, OAI-RAN implements three algorithms: a channel-aware proportional fairness algorithm commonly used in commercial cellular networks, a greedy and fair Round Robin scheduling algorithms and the last it aims to maximize the throughput.

It is possible to interface the eNB with both commercial or open source EPCs, as well as different SDRs. Moreover, both Commercial Off-the-Shelfs (COTSs) devices and SDRs can be used as UEs.

OAI-RAN also includes two simulation frameworks [18]. The first is the RFsimulator that allows all the tests without a RF board by replacing the radio board with a software (TCP/IP) communication. The OAI-RAN and the OAI UE can communicate as if a RF interface was present between them, but without any real-time clock constraints. Also MIMO is supported by the RFsimulator.
The second simulation framework is a Layer 2 simulator which implements Layer 2 and Layer 3 functionalities only, without the need to interface with any external SDR device. Being transparent to Layer 1 procedures, this simulation environment provides a useful tool to evaluate the performance of algorithms and protocols at the upper-layers.
Using actual radios or even the RFsimulator does not allow testing a large number of UEs. Therefore, the L2simulator enables the connection between OAI UE and OAI eNB/gNB through the nFAPI interface defined by the Small Cells Forum (SCF). By sending nFAPI messages between the eNB and OAI UE, the PHY layer of OAI is entirely avoid. Normally these messages are sent from the MAC to PHY layer for further processing.
The OAI L2-Simulator is a flexible tool that can be useful for many purposes, especially testing the network with a large number of UEs.

4.1.2 Description of OAI-CN features

The other key element of an LTE network is the core which is implemented in the OAI-CN framework.

As regard for the fundamental functions implemented from OAI-CN, are included Network Access Control Functions, such as authentication and authorization, admission control, policy and charging enforcement. Then, packet routing and transfer functions, like IP header compression function and packet screening. Further, there are Mobility Management Functions, which includes functions as

reachability management for UE in ECM IDLE state, and security functions and radio resource management functions as well. Finally, OAI-CN implements Network Management Functions, such as: GTP C signaling based Load and Overload Control, load balancing between MME instances, MME control of overload and PDN Gateway control of overload.

Mosaic5G leverages on OAI-CN, in particular the second version (v2), which is based on the latest OAI-CN version (develop branch), the one which implements the Control and User plane separation (CUPS) of EPC nodes. CUPS is one of the main novelty of 3GPP Release 14, and represents an architectural enhancement feature that introduces the concept of separation between CP and UP of EPC nodes, such as SGW, PGW, Traffic Detection Function (TDF).
The division of control and data plane functions of PGW and SGW into different entities leads the service providers to gain a greater flexibility in dealing with UP latency. That introduces a new protocol, named Packet Forwarding Control Plane (PFCP), as the CP-UP interface standard.
CUPS is not a new concept in the wireless world and it has quickly becoming an integral part of the 5G network development. It allows operators to separate the EPC into a CP that can sit in a central location, while the UP can be placed closer to the application it is supporting, to reduce latency and saves resources.
Therefore, SPGW is composed of almost 2 network functions: SPGW-C and SPGW-U.

The SGW and PGW entities carry packets through the GPRS Tunnelling Protocol (GTP) for both the user and the control planes. This is done done thanks to the GPRS Tunneling Protocol User Plane (GTP-U) and the GPRS Tunneling Protocol Control Plane (GTP-C), which exploit UDP as transport protocol.

As regard for the 3GPP interface implemented by the software, the most relevant are:

- S1-MME: It allows the flow of S1-AP control application protocol between E-UTRAN and MME.
- S1-U: It enables the flow of user plane data between E-UTRAN and SGW.
- S6a: It links the MME and the HSS, allowing user authentication and authorization, and the transfer of user subscription.
- S10: It allows the control among different MMEs.
- S11: It is a control plane interface between MME and SGW used to manage the Evolved Packet System (EPS).

- SGi: It enables the connection between the PGW with the Internet.

Concerning the SW and HW requirements to deploy OAI-CN framework, the target Operative Systems are Ubuntu 18.04 (bionic) server edition and Red Hat Enterprise Linux 8. As for the hardware characteristics it is required a CPU x86-64 Intel/AMD, while a generic kernel is enough for Linux. Moreover, the deployment is feasible on PC, servers, containers or Virtual Machines.

4.2 FlexRAN

Mosaic5G offers a solution to provide RAN sharing among service providers, while guaranteeing isolation and customization. The framework is called FlexRAN [19], [10], and it is an open-source implementation of a flexible and programmable platform for a Software-Defined Radio Access Networks.
Software-defined radio access networking (SD-RAN) is the main enabler for RAN slicing: it permits the decoupling of control and user plane (CP/UP) to control, program and coordinate multiple nodes through a central controller. Up to now, SD-RAN offered programmability only through a re-configuration of the network. However, with a view to enabling the service-oriented vision of next generation, it is fundamental extend the capabilities of the control plane in order to satisfy the requirements of each service, similarly to the idea behind network function virtualization (NFV). Furthermore, making smarter the RAN through data-driven control.

The FlexRAN platform is constituted by two main components, the FlexRAN Service and Control Plane and FlexRAN Application plane and it decouples the RAN control and data planes through a new, custom-tailored southbound API.

The FlexRAN service and control plane follows a hierarchical design which is composed of a Real-time Controller (RTC) that is connected to a number of underlying RAN runtime, one for each RAN module. The separation among control and data plane is supplied by RAN runtime environment which acts as an abstraction layer with RAN module on one side and RTC and control apps on the other side. The presence of FlexRAN protocol allows the communication between RTC and the RAN agent embedded in runtime environment .
RAN control applications, that allow to monitor, control and coordinate the state of RAN infrastructure, can be developed both on the top of the RAN runtime and RTC SDK. These applications range from soft real-time application such as monitoring through statistics reporting, to more sophisticated distributed applications which can modify the state of the RAN in real-time, like for example the MAC scheduler. Thanks to the software-defined RAN controller, slices can be dynamically added, removed or modified through a slice management module in the RAN controller

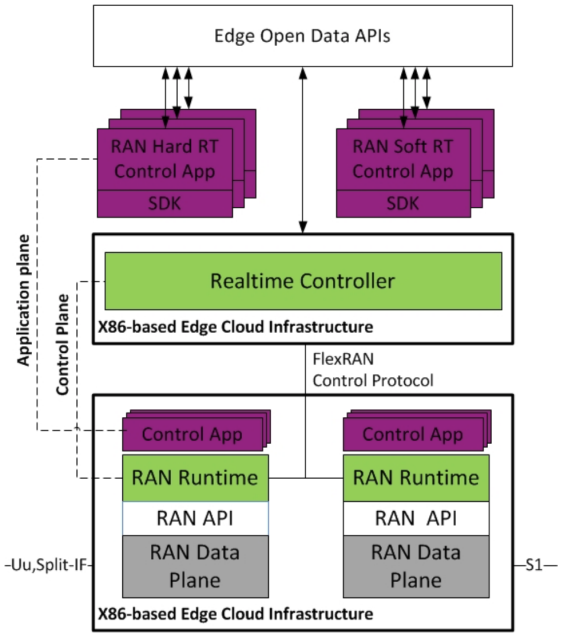

Figure 4.5: FlexRAN architecture.

agent. Furthermore, it is allowed to dynamically change parameters of a slice or its scheduler. The controller can modify "on-the-fly" the slice scheduling algorithm in order to fulfill all the slices' service level agreements (SLA).

4.2.1 Description of FlexRAN features

The separation of RAN control and data plane provided by FlexRAN, leads to several benefits, such as the reduction of the complexity when developing new control solutions and allowing operators to open their RAN service environment to authorized third-parties, promoting openness and innovation which can lead to fast deployment of innovative applications and services for verticals.

The control plane is placed into a single logically centralized controller, facilitating the coordination between base stations and simplifying the development of advanced control applications. Moreover, FlexRAN supports the deployment of real-time control applications, like MAC schedulers, with very stringent time constraints.

FlexRAN offers a RAN infrastructure flexible and programmable thanks to the introduction of RAN API and virtualized control functions, which are responsible of BS control operations. This allows part of the system's logic to be easily modified by replacing or extending the corresponding function without affecting the rest of

the system.
The virtualized control functions of FlexRAN are exploited through a set of mechanisms of delegation of control functions from the master controller to the BS at runtime which simplify the reconfiguration on-the-fly of parameters. This feature makes the system very flexible and adaptable to the underlying networking conditions and to the specific requirements of the network operator.

4.2.2 Software implementation characteristics

The FlexRAN real-time controller, the so called Master, was developed in C++ and currently supports x64 Linux systems. The implementation supports both hard and soft real-time mode of operation to support different critical issues related to time of applications and the requirements of the network operators and service providers. The FlexRAN real-time controller is released under a MIT License.

The FlexRAN Runtime, the Agent, was developed in C on top of OAI LTE open-source platform and has already been integrated to the current version of the project. It supports execution environment for local RAN control applications. The FlexRAN Runtime is released under OAI Public License V1.0 [19].

Chapter 5

Description of testbed deployment

The proposed architecture has been deployed through the Mosaic5G architecture, which relies on FlexRAN built above the OAI's open network architecture. At the moment of writing, as it has been explained in the previous chapters, Mosaic5G ecosystem did not released yet 5GC and NR frameworks supporting slicing features. Therefore, the testbed has been deployed with one EPC and one eNB, and the resource control is extended only at the RAN domain. As soon as a complete 5G open source network will be released in the future, it could be used to verify the concept of network slicing under this architecture.

The testbed shown in Fig. 5.1 employs two Linux-based PCs equipped with i7-8750H CPU at 4.1 GHz. These have been connected each other through an Ethernet cable. Both the Operative Systems were running Ubuntu. More precisely, on one PC has been implemented the LTE Core Network and on the other, the RAN part and the Users Equipment in simulation mode (i.e. Layer 2 Simulator). The choice of making use of a simulator instead of real Base Station (BS) and UEs, comes from the necessity to analyse dense networks with a reasonable low budget for hardware.
Furthermore, in the eNodeB, it has been implemented the two-level scheduler. The eNodeB is controlled and configured through the FlexRAN protocol. The SO is run as an application on top of the eNB controller: it allows the slice resource manager (SRM) to ensure intra-slice traffic scheduling and the resource mapper (RM) to assign PRBs to UEs according to the mapping provided by the SRM.

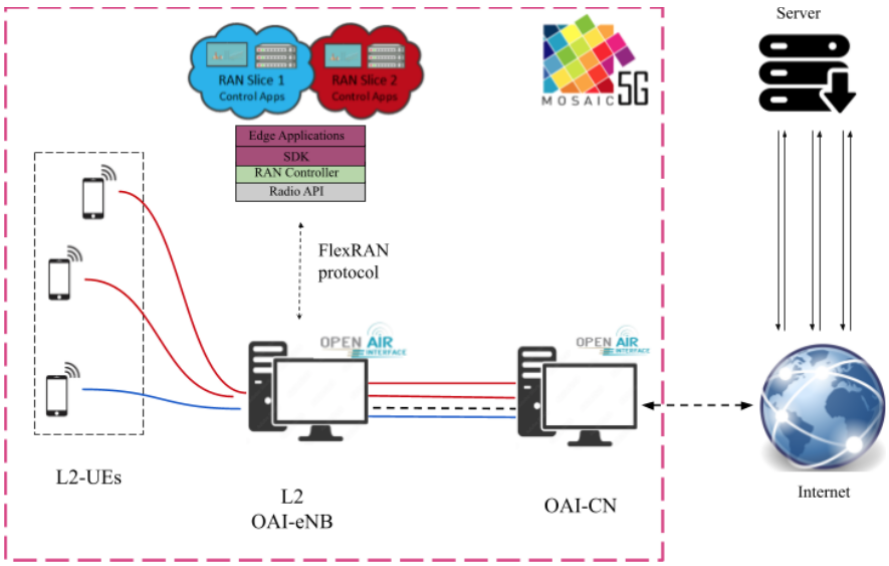

Figure 5.1: Testbed set-up.

5.1 Core Network

As for the OAI Core Network, the version exploited in this work has been the second (OAI-CN v.2), which is based on the latest developed branch. In addiction, OAI implements a CN such as the serving gateway (SGW) is deployed within the PDN gateway (PGW), therefore there is no S5 and they are considered as one entity. The main innovation introduced by the second version OAI-CN, compared with the previous, is the CUPS (Control and User Plane Separation) implementation. Hence, the decoupling between Control and User Plane functionalities.

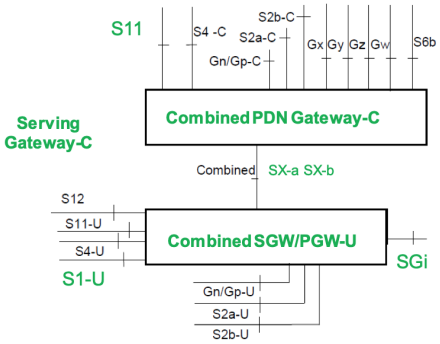

Figure 5.2: Architecture model with CUPS for a combined SGW/PGW [20].

Since the presence of Layer 2 Simulator creates some problems, in order to deploy a mobile system where users can exchange data without bypassing the LTE network, the CN requires to be deployed on a separate host from L2 simulator.

Reviewing all the elements which compose the EPC, to instantiate a working OAI-CN it results that the configuration of some of them have to be modified, while others are connected with already operative loopback interfaces. In particular, since CN and RAN modules have been deployed on different machines, after the installation of the OAI-CN and Layer 2 Simulator frameworks, it is necessary to modify the interfaces according to the Ethernet ones.

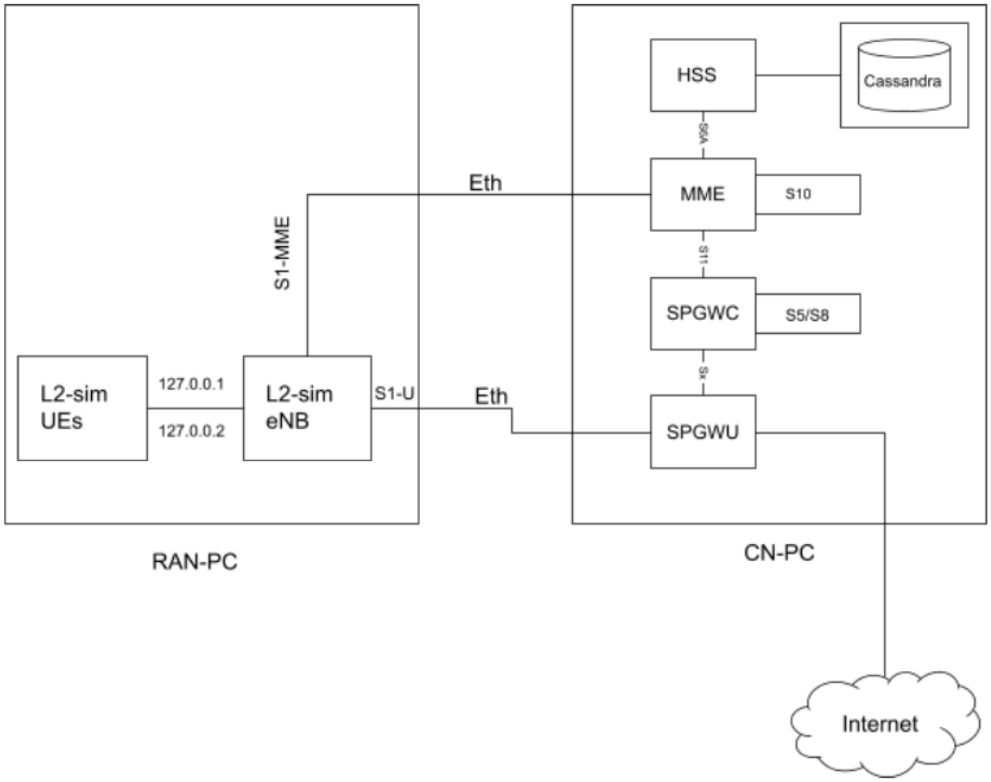

Figure 5.3: Block-scheme testbed architecture.

Firstly, the OAI-HSS database, which is based on Cassandra DB and employs docker to facilitate the installation and the deployment procedure, is connected to MME with a loopback interface already configured.
As regard MME entity, it is fundamental to modify in the related configuration file, the interface assigned to S1-MME channel. Its IP address will be 170.16.1.70/24. Similarly, for the S1U channel which connects SPGWU with eNB, the new IP

address will be 170.16.1.70/24.
The last interface which requires to be changed is the SGi, which links the gateway to the Internet world with the address 10.0.0.15/24.
As prelude before, Sx, S6a, S10,S11, SGW, PGW are internal interface to the CN branch. Hence, the addresses are the default ones, related to the internal network and assigned to the loopback interfaces.

The testbed requires to be associated with a specific network, identified with the Public Land Mobile Network Identifier (PLMN). PLMN is a number which identifies a combination of wireless communication services offered by a specific operator in a given country. A PLMN typically includes several cellular technologies offered by a single operator within a country, generally called cellular network. A PLMN is composed by the Mobile Country Code (MCC) and the Mobile Network Code (MNC). The MCC is assigned by ITU, while the MNC is assigned by the National Authority. Substantially, the MCC and MNC are used in order to uniquely identify a mobile subscriber's network in a specific country.
Therefore in MME module it necessary to set the proper Globally Unique MME Identity (GUMMEI) and Tracking Area Identity (TAI) lists. GUMMEI is the component within the Globally Unique Temporary Identity (GUTI) that uniquely identifies the MME and TAI is the identity used to identify tracking areas. TAI is constructed from the MCC, MNC and TAC (Tracking Area Code).
In this case study it has been exploited an MCC equal to 208 and a MNC of 95.

5.2 Radio Access Network

For testing purposes, the Radio Access Network has been deployed without a physical SDR device, but exploiting the L2 simulator provided by OAI, which emulates the eNB and UE.
The L2 simulator completely avoids the Physical layer of OAI by sending network Functional Application Platform Interface (nFAPI) messages between eNB and OAI UE. Normally, these messages are sent from the MAC to the PHY layer for further processing. Thanks to the L2 emulator, the eNB and UEs run the full LTE stack except the PHY layer, allowing to support a larger number of UEs for scalability purposes, and providing a more stable setup. The snap is based on the branch mosaic5g-oai-sim.
The nFAPI allows the functional split between the MAC and PHY functions that enables virtualization of the MAC function. The motivation to the usage of nFAPI is to encourage competition and innovation among suppliers of platform hardware, platform software and application software by providing a common API around which suppliers of each component can compete.

Moreover, the UE executable is able to "simulate" multiple UEs in order to stimulate the scheduler in the eNB. The reasons behind the choice of using a simulator of second layer are supporting multiple UEs.

As introduced in core section, eNB should be edited to match the network configuration as well. Especially, it requires the configuration of two interfaces, to MME and to SPGW. The first is S1-C is the control interface to exchange message with MME, the latter is S1-U, a data plane interface for data packets.
Therefore, the configuration file of L2 simulator eNB (eNB.l2sim.conf), has been modified to match the set up as follow.

Firstly, it is needed the adjustment of ipv4 in section "MME parameters", substituting this field with MME IP-address used in Core Network.

```
    ////////// MME parameters:
mme_ip_address = (
  {
    ipv4       = "172.16.1.70"; //
    ipv6       = "192:168:30::17";
    active     = "yes";
    preference = "ipv4";
  }
);
```

At eNB side, S1-MME is associated with the interface assigned to the cable, with address 172.16.1.80/24 , the same for the S1U . Consequently, in "Network Interfaces" section, these fields have been edited accordingly.

```
NETWORK_INTERFACES :
{
    eNB_INTERFACE_NAME_FOR_S1_MME   = "enx0000100032f0"; //
    eNB_IPV4_ADDRESS_FOR_S1_MME     = "172.16.1.80/24"; //
    eNB_INTERFACE_NAME_FOR_S1U      = "enx0000100032f0"; //
    eNB_IPV4_ADDRESS_FOR_S1U        = "172.16.1.80/24"; //
    eNB_PORT_FOR_S1U                = 2152; # Spec 2152
    eNB_IPV4_ADDRESS_FOR_X2C        = "127.0.16.3/24";
    eNB_PORT_FOR_X2C                = 36422; # Spec 36422
};
```

The communication among UEs and eNB in simulation environment, happens through a tunnel created with the loopback interfaces 127.0.16.1 and 127.0.16.2

with 255.0.0.0 netmask. Therefore, in the same configuration file, in MACRLC section, `remote_s_address` and `local_s_address` have been changed.

```
    MACRLCs = ({
  num_cc = 1;
  local_s_if_name  = "lo";
  remote_s_address = "127.0.16.1"; //
  local_s_address  = "127.0.16.2"; //
  local_s_portc    = 50001;
  remote_s_portc   = 50000;
  local_s_portd    = 50011;
  remote_s_portd   = 50010;
  tr_s_preference  = "nfapi";
  tr_n_preference  = "local_RRC";
});
```

Also the network configuration should be properly edited to be compliant among CN, eNB and UEs.

5.3 User Equipment

Last but not least, the UE configuration file should be edited as well. Indeed, in *ue.l2sim.conf*, remote and local addresses, related to the tunnel associated with eNB, have to be changed to match the eNB configuration.

```
L1s = (
        {
        num_cc = 1;
        tr_n_preference = "nfapi";
        local_n_if_name  = "lo";
        remote_n_address = "127.0.16.2"; //
        local_n_address  = "127.0.16.1"; //
        local_n_portc    = 50000;
        remote_n_portc   = 50001;
        local_n_portd    = 50010;
        remote_n_portd   = 50011;
        }
);
```

The final purpose of deploying an EPC is to connect a 4G UEs to internet through an eNB.
A Cellular Network has a very complicated structure and it is made up of many different layers. In addition, the network is communicating with many different users (UEs) at the same time. Therefore, it is needed to configure unique users IDs which include different fields.
When "burning" a simcard, it is important to decide some parameters, such as International Mobile Subscriber Identity (IMSI), LTE_KEY and OPC_KEY and more.
IMSI is a number that uniquely identifies every user of a cellular network. It is stored as a 64-bit field and is sent by the mobile device to the network. It is also used for acquiring other details of the mobile in the home location register (HLR) or as locally copied in the visitor location register. Going deeper, the parameters that have to be configured are the PLMN, and the SIM details, such as MSIN, USIM_API_K, OPC.

It should be provided a list of the known PLMNs, with the following parameters:

```
  PLMN0: {
       FULLNAME="Test network";
       SHORTNAME="OAI4G";
       MNC="95";
       MCC="208";

  };
```

and for each UE, its characteristics.

```
  UE0:
{
  USER: {
      IMEI="356113022094149";
      MANUFACTURER="EURECOM";
      MODEL="LTE Android PC";
      PIN="0000";
  };

  SIM: {
      MSIN="0000000001";
      USIM_API_K="8BAF473F2F8FD09487CCCBD7097C6862"
      OPC="8E27B6AF0E692E750F32667A3B14605D";
      MSISDN="33600000001";
  };
```

```
    # Home PLMN Selector with Access Technology
    HPLMN= "20895";

    # User controlled PLMN Selector with Access Technology
    UCPLMN_LIST = ();

    # Operator PLMN List
    OPLMN_LIST = ("20895");

    # Operator controlled PLMN Selector with Access Technology
    OCPLMN_LIST = ("20895");

    # Forbidden plmns
    FPLMN_LIST = ();

    # List of Equivalent HPLMNs
    EHPLMN_LIST= ();
};
```

5.4 FlexRAN

After the deployment of CN, eNB and multiple UEs through Layer 2 Simulator, it is required to install the RAN controller. FlexRAN Real time Controller is then installed from the Mosaic5G builder and properly configured.

5.4.1 Initialization

In order to be able to run FlexRAN, the master controller node should be initialized and some permissions have to be granted (see flexran.info):

```
$ sudo snap connect flexran:log-observe

$ sudo snap connect flexran:process-control
```

Then, the controller has to be enabled also in the RAN's configuration, setting the parameter `"FLEXRAN_ENABLED"` to `"yes"`. Moreover, also the IP address need to be set correctly. In this study case, since the RAN controller has been deployed on the same machine as for the RAN, the IP address is the local one.

```
    NETWORK_CONTROLLER: {
    FLEXRAN_ENABLED        = "yes";
    FLEXRAN_INTERFACE_NAME = "lo";
    FLEXRAN_IPV4_ADDRESS   = "127.0.0.1";
    [...]
}
```

5.5 Slice creation

In order to deploy a network which allows slicing management at the RAN side, the current testbed utilizes FlexRAN Controller. Such an application allows a flexible and programmable control of the underlying RAN infrastructure, thanks to the incorporation of specific APIs and the virtualization of some control functions.

One of the innovating feature that FlexRAN made it available, is the slicing concept at the RAN side.
Through an API endpoint it is possible to create a new slice configuration and to post it to the underlying agent, as a JSON file.
Therefore, the RAN sharing/slicing policy needs to be defined, and subsequently sends to the controller through the REST API. This consists in creating a JSON configuration file `ran-sharing.json`, defining a radio resource management policy. The JSON file includes a set of parameters that should be specified in order to define the slices.
Firstly, it is required to select for the Up Link or the Down Link section. Once decided the transmission direction to operate with, the selection of the "algorithm" filed includes "None" or "Static", as parameters. When selecting the first, only one single slice is created, without any resource separation. There is also the possibility to choose the algorithm followed by the entity that handles the resources, the so-called scheduler. In "None" mode only one scheduler manages all the resources

The related JSON file created should be the following.

```
    {
  "ul": {
    "algorithm": "None",
    "scheduler": "Round_robin_ul"
  },
  "dl": {
    "algorithm": "None",
    "scheduler": "Round_robin_dl"
  }
}
```

On the other hand, choosing "Static" as algorithm, allows to subdivide the physical resources in multiple sections, each of them independent and isolated from the other ones.

```
  {
"ul": {
  "algorithm": "Static",
  "slices": [
    {
      "id": 0,
      "label": "default",
      "static": {
        "posLow": 1,
        "posHigh": 12
      }
    },
    {
      "id": 2,
      "label": "two",
      "static": {
        "posLow": 13,
        "posHigh": 23
      }
    }
  ]
},
"dl": {
  "algorithm": "Static",
  "slices": [
    {
      "id": 0,
      "label": "default",
      "static": {
        "posLow": 0,
        "posHigh": 5
      }
    },
    {
      "id": 2,
      "label": "two",
      "static": {
```

```
          "posLow": 6,
          "posHigh": 12
        }
      }
    ]
  }
}
```

It is necessary to emphasize that the scheduler (within UL/DL) and the slices parameters are mutually exclusive, since the first is applied only if no slicing algorithm is used.
Whenever a slicing configuration is used, some parameters specifying the characteristics of the slice have to be set.
It is notable that DownLink and Uplink slicing are completely independent from each other. Indeed, it is possible to have just a DL slice without the corresponding UL slice, or a large DL slice at the beginning of the RB spectrum with a small corresponding UL slice at the end of the spectrum.

Moreover, the allocation of the Resource Block of slicing is completely free within the boundaries of LTE (i.e. respecting some fundamental thing such as the RBGs in DL.)
Everything is subject to the physical layer restrictions of LTE, which in particular means that a UE still has to observe the complete bandwidth for control information, and it is impossible to modify the actual frequency allocation, but only resource mapping.
The current 4G MAC slicing strategies only influence, where and how MAC resources are allocated to (groups of) users. In particular, it does not change any physical layer parameters, because LTE does not support this. Therefore, no matter what slicing strategy has been chosen, the duplex spacing is defined by the bandwidth configuration, which is not related to slicing.

Although, what it is possible to change for slicing configuration is which RBs are eligible for certain user groups and with which scheduler they will be assigned to the users. Those are the fundamental parameters to be tuned by the SO to change the slices accordingly with the traffic requirements.

5.5.1 Physical Resource Block

The FlexRAN Controller provides only static slicing algorithms, meaning that each slice configured cannot be dynamically shared. The physical resources assigned

to each sub-network are isolated and uniquely exploited by the group of UEs associated to them.
A resource block (RB) is the smallest unit of resources that can be allocated to a user. The dimension of a RB is 180 kHz wide in frequency and 1 slot long in time. In frequency, resource blocks is typically large 12 x 15 kHz subcarriers in an interval of 1 slot (0.5 ms). For the majority of channels, the number of subcarriers used per resource block is 12. In LTE the bandwidth is fixed to 180 kHz, since each sub-carrier has 15 kHz, different from the 5G in which the sub-carrier spacing varies.

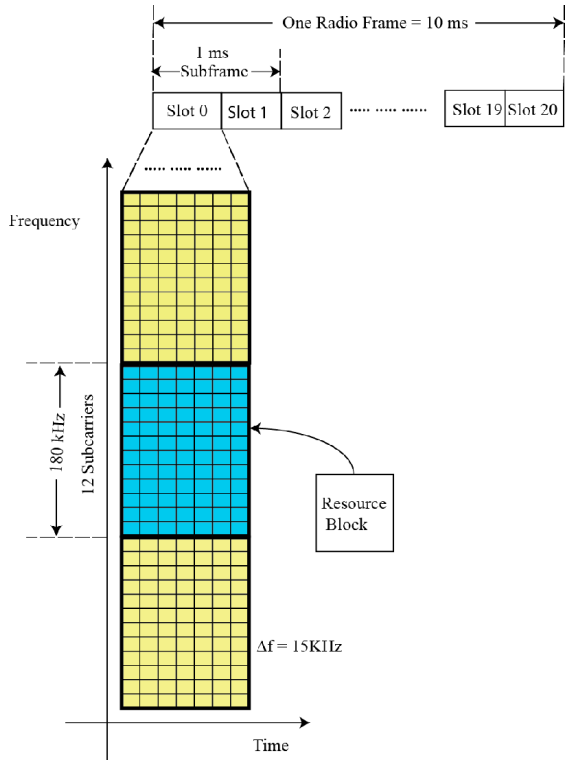

Figure 5.4: LTE Resource Block structure in time and frequency domain.

Frequency units can be expressed in number of subcarriers or resource blocks. The bandwidths defined by the standard are 1.4, 3, 5, 10, 15, and 20 MHz. The table below shows how many subcarriers and resource blocks are present in each bandwidth for Uplink and Downlink.

Bandwidth	Resource Blocks	Subcarriers (downlink)	Subcarriers (uplink)
1.4 MHz	6	73	72
3 MHz	15	181	180
5 MHz	25	301	300
10 MHz	50	601	600
15 MHz	75	901	900
20 MHz	100	1201	1200

Figure 5.5: Bandwidth and RBs

In FDD (Frequency Division Duplex) mode, UL and DL frames are both 10 ms long and are separated both in frequency and time.

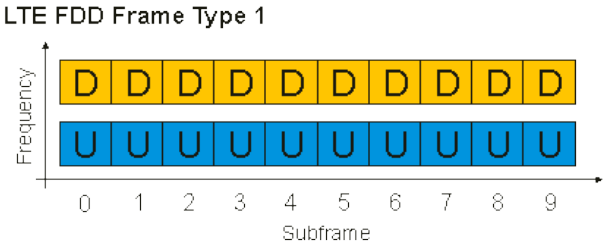

Figure 5.6: LTE FDD frame

Resource Allocation Type specifies the modality in which the scheduler allocates resource blocks for each transmission. Since to give the maximum flexibility of resource block allocation, it would create too much complicated situations, LTE standardized some resource allocation types that use predefined procedures. In particular the resource allocation types in LTE are three: Type 0, 1, 2. The OAI testbed uses resource allocation type 0 for slicing (i.e. bitmaps).

Resource Allocation Type 0 is the simplest way of allocation resources. First it divides resource blocks into multiples of groups. These groups of blocks are called Resource Block Group (RGB). The number of resource block in each group varies depending on the system bandwidth. It means RBG size gets different depending on the system bandwidth used. The relationship between RBS size, that is the number of resource block in a RBG, and the system bandwidth behaves as follows.

System BW	RGB size
1.4	1
3	2
5	2
10	3
15	4
20	4

Table 5.1: Resource Allocation Type 0

Therefore, since the testbed is constrained by the limits of the L2 simulator which allows a bandwidth of 5 MHz, the experiments will be based on a system of 25 PRBs in UpLink. That corresponds, as shown in the table above, to a range of 0 to 12 RGBs in Down Link (the last RBG has only one RB).

Going into detail of slice creation, it has remarked that the allocation of RB is one of the major parameters to be tuned to create customized portion of the network on the same physical infrastructure. Therefore, in the JSON file for slice configuration, the fields posLow and posHigh are numbers intended in terms of resource block groups (RBG) in DownLink and resource block (RB) in UpLink. Especially, posLow is the lower (inclusive) starting resource block group for a slice, while posHigh is the upper (inclusive) starting resource block group for the same slice. Both of them should not overlap with any other existing or new slice. Those indexes can assume different values, according to the bandwidth considered by the system, as shown in table 5.4
Furthermore, it is not checked that the channel bandwidth can actually accommodate the slice posLow=100 and posHigh=110. They are valid entry, but never work, since LTE has a maximum bandwidth of 100 RBs.

It is also important to note that OAI reserves the first and the last 1,2 or 3 RBs, for bandwidths 25, 50 or 100 RBs, respectively, for PUCCH. PUCCH is the Physical Uplink Control Channel and carries UpLink control information, such as channel quality information, acknowledgements and scheduling requests.

Last but not least, with regard to the previous JSON slice configuration file, since the L2 simulator is based on an LTE configuration of 25 RBs, and which is not shared with other slices, that file defines two active slices with IDs 0 and 3, each of them with roughly 50% of resources.

5.5.2 Slicing Scheduler

The other important parameter to be configured during the slice creation is the scheduler associated to every slice. It is independent and can be different for any sub-network. Currently, the scheduling strategies implemented and available to testing are Round Robin, Proportional Fair, and Maximum Throughput.

Scheduling is the process of allocating resources in UL and DL for transmitting data. A scheduler is responsible for resource blocks (RB) allocation among cellular users according to their requests. MAC scheduling is one of the most important function of eNB, as well as one of the most challenging due to the latency requirements. According to the scheduling algorithm strategies and some parameters (i.e. information link state, number of sessions, state of the queue etc.), the scheduler allocates the common resources to the users every transmission time interval (TTI). Scheduling aims at optimizing radio resource utilization over time and frequency domain in order to deliver the best possible user experience based on different criteria. Therefore, scheduling algorithms play an important role to guarantee quality of service (QoS) parameters in wireless links.

In the RAN standard there is not a specific scheduling algorithm. Since the access network has the challenging role of dealing with real-time radio resource allocation serving different users with diverse requirements, the algorithm implementation is standardized but depends on network requirements.

Another major contributor is the CQI (Channel Quality Indicator) reported by every UE, which is very helping for the eNB to estimate the Downlink channel quality.
Therefore, every UE sends to the eNB an indicator, raging from 0 to 15, which indicates the quality of the channel based on the Signal-To-Noise Ratio (SNR). The CQI symbol, which is a measurement of the channel status reported by every mobile user based on its past experience, changes along time according to the channel state.

The CQI provides an estimate of the highest modulation-and-coding scheme that, if used carefully, would result in a smaller block-error probability. The reason behind the use of CQI as a feedback quantity instead of, for example, the SNR directly, is to account for different receiver implementations.

The scheduler algorithms available in FlexRAN are:

- Round Robin (RR);
- Proportional Fair;
- Maximum throughput;

1. Round Robin (RR): Round Robin is one of the most popular scheduling algorithms in scenarios where multiple agents share a common resource, like bandwidth. It is applied in a wide variety of environments, including bandwidth allocation in LAN and mobile networks. Round Robin assigns short time slices to the users in equal portions and in circular order. Hence, the buffer is organized in separate queues, served in FIFO manner. These time slices are typically very short; as such Round Robin scheduling can be alternatively considered as an equal splitting of the resource among all active jobs. The advantages of RR are that is instantaneously fair, it assigns all active agents an equal amount of the resource at any given moment and it is starvation-free [21].
   RR algorithm gives equal scheduling chance to each user in the cell. The scheduler assigns resources cyclically to the users without taking channel conditions into account. This is a simple procedure giving the best fairness when the packet size is constant. Although for variable packet size, the scheduler before switching to the next queue need to wait the current packet transmission to finish. This translates into a greater global channel occupancy for larger packets. In addiction, it does not take care of channel condition, offering poor performance in terms of cell throughput.

2. Proportional Fair (PF): Proportional-fair scheduling is a compromise-based scheduling algorithm. Its goal it maintaining a balance between two competing interests: maximize the total throughput of the network while at the same time allowing all users at least a minimal level of service. This is done by assigning each data flow a data rate or a scheduling priority (depending on the implementation) that is inversely proportional to its anticipated resource consumption. Therefore, PF scheduling algorithm assigns radio resource to user with highest instantaneous achievable data rate relative to its past average data rate. The scheduler can exercise Proportional Fair (PF) scheduling allocating more resources to a user with relatively better channel quality or to the user with the lower throughput, so even if a UE has a poor CQI it will get resource. This offers high cell throughput as well as fairness satisfactorily. Thus, Proportional Fair (PF) scheduling may be the best option.

3. Maximum Throughput: the scheduler assigns resources to the user with the best channel quality. This offers excellent cell throughput but it is not fair.

For instance, a way to schedule data transfer is through the use of prioritization coefficients. Here, it is scheduled the channel for the UE that has the maximum of the priority function:

$$P = \frac{T^{\alpha}}{R^{\beta}} \tag{5.1}$$

Where T denotes the potentially achievable data rate for the user in the present time slot, R represents its past average data rate. α and β are the tuning parameters, to adjust the "fairness" of the scheduler.
Tuning α and β, it is possible to find a balance between serving the best mobiles (the ones in the best channel conditions) and serving the costly mobiles often enough to guarantee an acceptable level of performance and avoid starvation. Theoretically, the best mobiles should be served more often than the low rate users. Considering the parameters pattern ($\alpha = 0$ and $\beta = 1$) the scheduler acts in a Round Robin fashion and serves all mobiles equally often, with no regard for resource consumption. On the other hand, if ($\alpha = 1$ and $\beta = 0$) the scheduler will always serve the mobile with the best channel conditions (higher CQI). This will maximize the throughput of the channel, but mobiles with low data rate are not served at all. That is the way Maximum Throughput acts. Using $\alpha \simeq 1$ and $\beta \simeq 1$ will yield the Proportional Fair scheduling algorithm.

In some studies, it is found that proportional fair gives very good data rate in most cases. Although round robin gives better individual data rate when the UE is located too far away from eNodeB, the absolute value of this data rate is not that high. Thus, the resources are not then properly utilized and so, proportional fair may still be a better choice. Round robin treats the UE with the best fairness but proportional fair can maintain a balance between fairness and throughput.

So far, the scheduling algorithms implemented and available in Mosaic5G environment are just the three presented above, but with a view to the increasing demanding of networks specialization and selective traffic, many more strategies of allocation should be developed. Starting from a delay-based scheduler for Low Latency data, until some algorithms customized for the QoS requirements to be granted.
The functions implementing these algorithms are only 100-150 longs and it should not be difficult to modify them in order to change other parameter configurations, like limiting the MCS, or implement new strategies, to match the target.
For instance, a parameter that could be tuned is Modulation and Coding Scheme, MCS, index which is directly related with the Modulation scheme and Coding Rate. This parameter should have a strong impact on the throughput achieved.

5.6 Slice orchestration

The ran-sharing.json file has to be sent when the agent runs, and posted through the following command:

```
$ curl -X POST http://172.16.1.80:9999/slice/eNB/-1 --data-
```

```
binary@ran-sharing.json
```

where the IP address specifies the machine on which the eNB runs.

5.7 UE association

Furthermore, UEs have to be associated to a slice. Newly arriving users will automatically placed in slice 0.
Although, to put a UE with a given IMSI or RNTI in another slice, it is necessary to create a second JSON file ue-association.json. In this file it is important to specify the IMSI of the UE and the ID of the slice to associate it with. The IDs of the UL or DL slice can be different.

```
 {
  "ueConfig": [
    {
      "imsi": <replaceWithYourImsi>,
      "dlSliceId": 2,
      "ulSliceId": 2
    }
  ]
}
```

The ue-association.json file has to be sent when the agent runs, and posted through the following command:

```
$ curl -XPOST http://192.168.12.45:9999/ue_slice_assoc/eNB/
--data-binary @ue-association.json
```

5.8 Monitoring applications

In order to control the evolving situation and monitor the status of the eNB and the UEs, some applications have been exploited.
Mosaic5g makes available the Store, a platform which contains SDKs for facilitated interaction with the underlying controllers as well as some demo applications. The Store is entirely written in Python.
The SDK provides a test mode that enables the development of applications without the need of having a running RAN. For instance, pre-recorded data can be used to test an application and deploy it later.

Applications can be split into two main classes: technology-dependent or technology-agnostic.

An example of technology-agnostic application exploited in this work is Drone, because it only shows data from the controller.
Another important application exploited to monitoring the current status of the RAN and the UEs is the Stats call, an API that gets RAN statistics, also human-readable, every TTI.

5.8.1 Drone

The drone application is a web server that can show the status of the RAN graphically in a web browser. It can be used to display the base stations connected to FlexRAN, and all the UEs that are connected to each base station. The parameters of each entity, which are directly passed from the controller, can be examined. Furthermore, it interacts with FlexRAN through other applications that can be configured at run-time within the drone application.
From this application, it is further possible to observe run-time a graphical representation of the throughput trend for each user. Moreover, many characteristics can be check, like the slice ID associated to each UE and the number of RGB associated.
Whenever users connect to the BS, they appear in the dashboard showing some information, such as the IMSI or the IP address of the corresponding Core Network. All the arriving UE are automatically put into slice 0.

From within the Mosaic5G root folder, navigate to the drone folder and start it:

```
$ cd store/sdk/frontend/drone/
$ python drone.py --port=8088 --address=127.0.0.1
```

Where "port" and "address" specify the web server's listen port and address. Opening a browser and navigating to 127.0.0.1:8088 it should be possible to see the current topology.

The dashboard shows the RAN, FlexRAN is already connected to the BS. The Core Network is not shown, but it is visible in the "plmnID" section among the BS information, as for the slice algorithm. It is possible to load a static slicing algorithm which partitions the resources to groups of users. Moreover, there is the NetStore that provides RAN applications and a ping Helper that integrates ping utility in the dashboard. To conclude, Drone application is a very useful tool to monitor the status of the network, which offers a practical graphical interface to interact with the ecosystem.

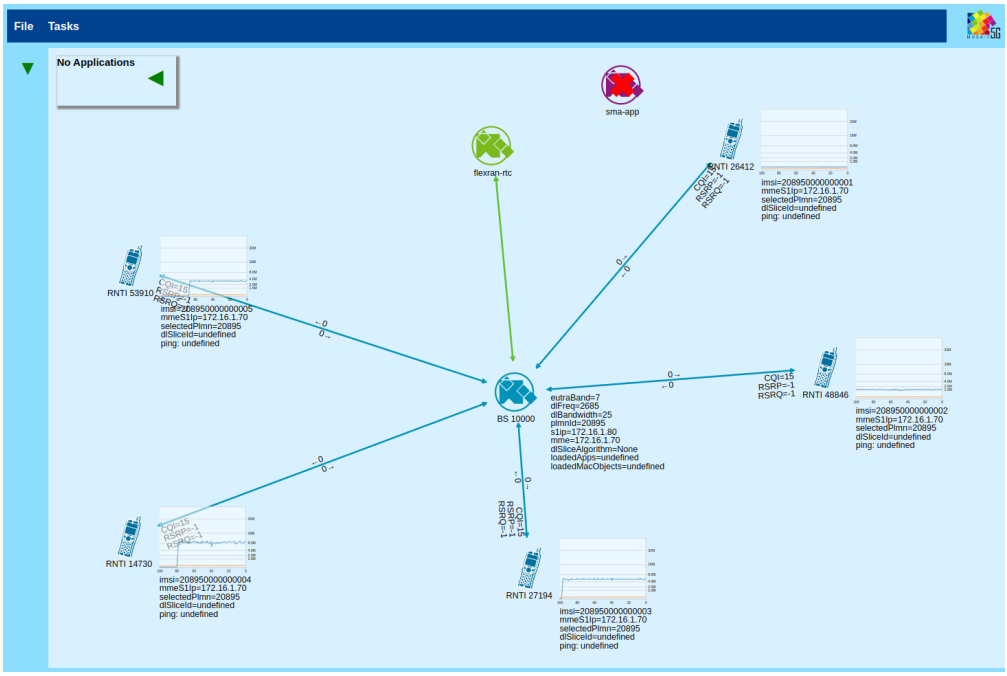

Figure 5.7: Drone Application

5.8.2 Statistics and reports

The FlexRAN controller provides also a northbound RESTful API for issuing control commands and for obtaining reports and statistics for base stations using simple HTTP requests.
The complete API documentation is available in the apidocs, at the following link: `https://mosaic5g.io/apidocs/flexran/`

Thanks to the FlexRAN northbound API, it is possible to get the RAN status and configuration of the current Transmission Time Interval (TTI), which is the time related to encapsulation of data from higher layers into frames for transmission on the radio link layer. TTI refers to the duration of a transmission on the radio link. That API allows retrieving the RAN condition for the current TTI related to all the eNBs connected to the controller. The output can be in a human-readable format or in JSON file.
In order to access to these statistics, it is simply necessary typing the command:

```
curl -XGET http://FlexRAN-URL:PORT/stats/conf/eNB/:id?
```

Where “id” is the ID of the desired BS. This can be one of the following: -1 (last added agent), the eNB ID (in hex, preceded by "0x", or decimal) or the internal agent ID which can be obtained through a stats call. Numbers smaller than 1000

are parsed as the agent ID. The predefined value is: -1.

In the specific testbed case, since the controller runs on the same machine of the eNB, it has been used the following:

```
curl -XGET http://127.0.0.1:9999/stats/conf/eNB/
```

To modify an existing configuration, use the endpoint:

```
curl -XPOST http://127.0.0.1:9999/stats/conf/eNB/
--data-binary @stats.json
```

There is only one field, the type of statistics to be returned. The type of statistics to be returned allowed are the following:

- `eNB_config`: static configuration (for eNB, UE, and LC)
- `mac_stats`: statistics about various eNB layers (PDCP, RLC, MAC)
- `all`: both of the above

The values allowed are `eNB_config`, `mac_stats`, `all`. Predefined value is `all`.

The status of the localhost reports a list of parameters, which includes:

- CQI information
- RRC measurements
- PDCP measurements
- MAC measurements
- GTP information

And the statistic are tracked continuously.
It is available a documentation for all end-points in JSON format, such as the capabilities of the controller, generated from source.
Giving a brief overview of the statistical reported, there are:

1. CQI information, where CQI stands for Channel Quality Indicator. It is an indicator carrying the information on how good/bad the communication channel quality is. CQI is the information that UE sends to the network.

2. The Radio Resource Control (RRC) measurements. RCC is a protocol used in UMTS, LTE and 5G on the Air interface. It is a Network Layer (L3) protocol used between UE and Base Station. The major functions performed by the RRC protocol consist in connection establishment and release, transport in broadcast system information, establish radio bearer, reconfigure, RRC connection mobility procedures, paging notification. In particular, between the statistics reported, there are the Primary Cell Reference Signal Received Power (RSRP) and the Primary Cell Reference Signal Received Quality (RSRQ). By means of the signalling functions the RRC can reconfigure UP and CP according to the network status and Radio Resource Management strategies to be implemented.

3. Packet Data Convergence Protocol (PDCP) measurements. PDCP is a protocol specified by 3GPP which located in the Radio Protocol Stack in the UMTS/LTE/5G air interface above the RLC layer. PDCP provides its services to the RRC and user plane upper layers. The following services are provided by PDCP to upper layers: transfer of user plane data, transfer of control plane data, header compression, ciphering and integrity protection. Therefore the measurements, includes a list of statistics on packet transmitted and received, sequence number, frame number, out-of-order packet, etc.

4. MAC measurements. They include several statistics information about scheduling, such as the number of PRBs exploited in UL/DL, or the transport block size. As well as, Modulation Coding Scheme (MCS) information.

5. GTP information. The status of the localhost reports a sequence of information related to the tunnelling protocol, such as tunnel-endpoint identifiers, or eNB and SGW IP addresses.

Moreover, it is present an API which allows to set and get the statistics configuration for a given Base Station. The configuration governs which set of statistics are sent and how often from the BS to the controller. The controller will delete all the current statistics configuration present in the BS and set the new given ones.

All the information about the specifics API can be found at the following page: `https://mosaic5g.io/apidocs/flexran/flexran_spec_v2.2.3.html#protocol.flex_dl_info`

Chapter 6

Simulations and results

The aim of the project focuses on the analysis of the latency performances in an heterogeneous network when exploiting Network Slicing technique at the RAN. More in details, the following paragraphs investigate the impact on Users behaviour under several network scenarios and with different NS configurations. Therefore, it has been carried out a survey on the Mosaic5G platform usage, understanding how to exploit the numerous parameters affecting the testbed performances. In particular, the main idea behind the simulations relies on the need of finding innovative solution to enhance mobile networks for supporting autonomous vehicles. To sustain the automotive industry, different categories of devices are required: some dedicated to constant monitoring, exchanging big quantities of data without particular constraints on latency and reliability and some related to mission-critical situations, driven by strict requirements on latency and accountability. In this respect, the simulations involve the coexistence of two services. As a general rule, improving the delay for prioritize services is the guiding thread of the tests proposed.

As mentioned above, the final goal is to study the problem of enabling the coexistence of multiple heterogeneous services within the same Radio Access Network (RAN) architecture, finding the benefits offered by a sliced RAN. In particular, the two services of interest for the next simulations are eMBB and URLLC. Typologies that have been largely discussed in Chapter 2.
In a nutshell, eMBB can be considered a direct extension of the 4G broadband service. It is characterized by large payloads and by a device activation pattern that remains stable over an extended time interval. The objective of the eMBB service is to maximize the data rate, while guaranteeing a moderate reliability. On the contrary, the URLLC service demands mission-critical, reliable communication, where a hard latency constraint must be fulfilled.
Service heterogeneity can be accommodated by network slicing,through which each service is allocated resources to provide performance guarantees and isolation from

the other services.

In order to simulate heterogeneous scenarios and test the variability of the delay in a multi-services network, a combination of network tools have been exploited. In particular, from a remote server under the same LAN it will be generated UDP traffic thanks to the Iperf3 tool. Different session of Down Link data will be started simultaneously, to contact all the mobiles.
Moreover, the users performances, affected by the scheduler and the resource association per typology of service, will be analyzed mainly in the shape of Round Trip Time (RTT), through Ping network tool.

Notably, the RTT is a measure of the time needed to send a signal and receive back the acknowledgment. Therefore, the measure of the delay by means of RTT, includes not only the scheduling time in Down Link, but also the Up Link ones. Anyway since in Up Link no slicing policy has been applied and the scheduler used is always the same for all the UEs, Round Robin, the waiting time for the UL queue should not be relevant. Moreover, all the simulations will be considered for Down Link data.
Round-trip time and Ping time are sometimes considered synonymous. Even if ping time can provide a good RTT estimate, there is a difference in how the test have been performed. Indeed, Ping tests are usually performed within a transport protocol that uses ICMP packets. While RTT is measured at the application layer (layer 7 of ISO/OSI) and includes additional processing delays caused by higher-level protocols and applications.

Given that the major point of interest will be the latency of the network caused by different slice configurations, it is interesting introduce this concept. Latency is the time required for a data packet to travel from the sending endpoint to the receiving endpoint (thus only one trip). And many factors may affect this path. Hence, network latency is closely related to RTT, but different.
Network latency is not definitely equal to half of the RTT, because it may be asymmetric between any two given endpoints. In addiction, RTT includes the processing delay at the echo endpoint.
However, considering the network in as steady as possible conditions, the RTT can be an acceptable measure of the slicing performances. In this sense, the ping measure has been computed between the mobile users and the Core Network, in order to avoid the Internet uncontrolled variables. Therefore, only the cable bandwidth and the processing time affect the measurements.

6.1 SWOT analysis of Mosaic5G framework

Whereas the progress of this project has experienced several problems, it has been fulfilled a SWOT analysis of the Mosaic5G framework based on the user expertise. A SWOT (Strengths, Weaknesses, Opportunities, and Threats) analysis, is a planning technique implied to help identifying benefits and drawbacks of a project. The SWOT analysis divides the examination in 4 categories, including strengths and weaknesses, opportunities and threats.

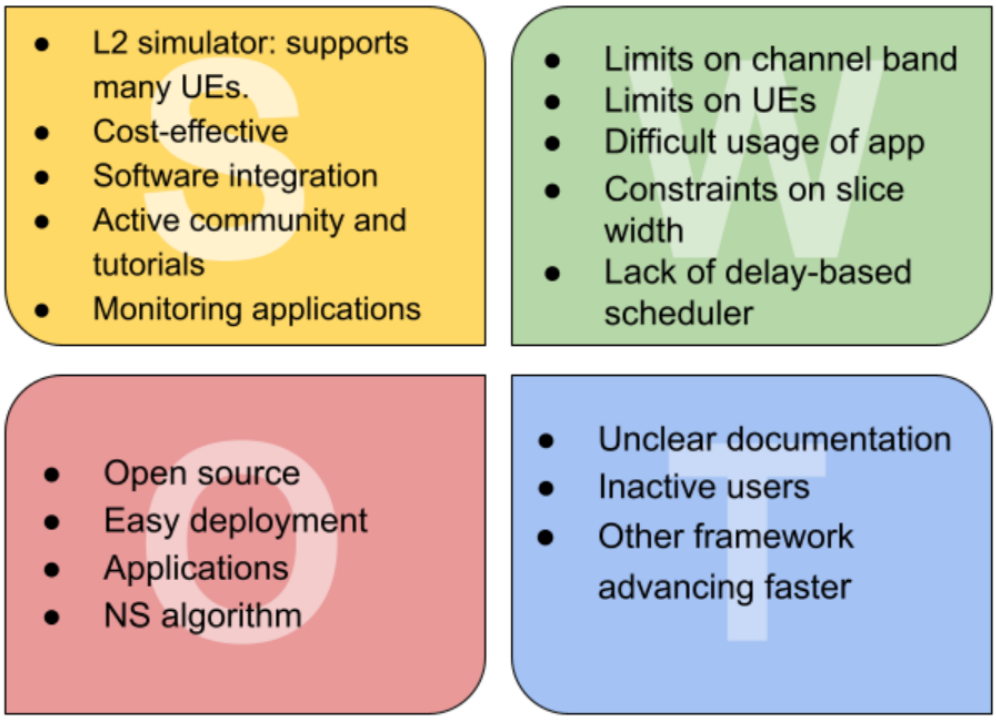

Figure 6.1: SWOT analysis of Mosaic5G testbed.

6.1.1 Strengths and Weaknesses

In order to critically evaluate benefits and drawbacks experienced during the utilization of Mosaic5G framework, the analysis starts focusing on strengths and weaknesses of the project. Concerning the keys strength are the possibility of substitute the SDR, Software Defined Radio, the channel and the users, thanks to the Layer 1 or Layer 2 simulators. Both of them allow to reduce the hardware costs, creating an efficient testbed to test applications over the network. Especially, the L2 simulator is particularly efficient from this thesis point of view, because it permits to reproduce a large number of UEs and test them all together.
Another point in its favor is the facility of deploying the testbed thanks to the snaps and to the software platforms integration already implemented. Therefore, the whole building procedure is strongly simplified, becoming just a matter of configuration. Moreover, the multiple projects, result of different alliances, are already merged together and totally compatible.
FlexRAN is a very powerful point of strength, allowing flexibility and easily controlling in real time the eNB. It allows to configure sub-networks runtime with simplicity, moving UEs on the fly from one slice to another. FlexRAN Controller leads to a flexible management of the physical resources, through API endpoint which communicate the configuration to the uderlying agent, in the form of JSON file. In addiction, also the Store should be highlighted, which contains SDKs for facilitate the interaction with the underlying controllers, through application such as Drone. Finally, another good point which could make prefer Mosaic5G framework on the other solutions is the presence of exhaustive tutorials and practical explanation on how to use the platforms. Last but not least, the presence of an active community and an efficient and quick mailing list support to clarify doubts, helps a lot to develop and interact with the platform.

On the other hand, there are several internal weak spots in Mosaic5G framework. First of all, the requirements on the CPU to build a performing testbed, which should not be underestimated to avoid risks of unexpected closure of the running processes. This kind of problem have been observed using the L2 simulator, whenever the traffic data started to be closer to the maximum channel capacity.
However the main lack of the simulator, considering the goals of the thesis, is the limit of the simulated channel bandwidth. Indeed, since the simulator is implemented with 25 Resource Blocks, the maximum throughput is 17/17.5 Mbps, which means that with 15 Mbps the channel is almost full. The problem is the inability of generating congestion situations to test the network underpressure, making it impossible to create the worst case avoiding the testbed collapse. Which, for simulation purposes, it is a big deal. Another issue is the difficulty in the usage of several simulated UEs simultaneously. The framework should provide up to 127

UEs, but actually it is really hard to test them all together. Even considering an aggregate throughput smaller than the total channel capacity, they are not able to be activated at the same time, unless with a amount of data considerably small. But then it is not really useful for experiments. Hence, relying on the experiences, the testbed works sufficiently well testing until 50/60 users simultaneously.
As regard for FlexRAN, the powerful of a real time and flexible controller for slicing purposes, is limited by the constraint of fixed sized Resource Blocks. A more advantageous system should be to adapt the size of the slice on the traffic demand. Consulting the website of the project Mosaic5G, it seems that similar algorithms, like the NVS (Network Virtualization Substrate) for an effective virtualization of wireless resources in cellular networks, are already being implemented but not yes available for the users. Furthermore, for the purpose of the work, which deals with the investigation of the situations where NS helps to reduce latency for mission-critical users, the absence of a delay-based scheduler implementation is a serious lack. Probably, handling UEs with a delay-based algorithm, could open the doors to clever and improved management solutions. Therefore, FlexRAN platform facilitates the development of control solution but at the time of writing, the capabilities are very limited.
Lastly, the Applications provided by the framework are several, but it misses a clear documentation on how to use them. Most of them are just mentioned and it is in not immediate understand their employment.

6.1.2 Opportunities and Threats

As regard for Opportunities, certainly the openness of the source code is a big advantages whether allows third parties contributions to enhance the project. Besides the code contribution, as well as an active community of users which share information each other about ideas, issues encountered and much more, could have a central role. Indeed, a good opportunity to highlight the potentiality of the Mosaic5G framework, would be a more interactive community, not only a mailing list. For instance, a channel where the users can share their works or discuss about the common problems. Furthermore, the simplicity of the deployment process by means of snaps and platforms already integrated, can be attractive for users and developers which can contribute to the enhancement of the platform. Another opportunity in this context would be the implementation of new algorithms for handle slices, not only the static one which creates too many limits.

Considering the threats, what can influence the success of the framework is the participation of the community, users and developers. In addiction the increasing hardware costs, which influence the industrial production of SDRs, by discouraging some users to deploy a testbed. Furthermore, another risk could be the lack of a

clear documentation which can dissuade people to make use of the platform. This is a matter not only of the features available, but also of tutorials, explanation and documentation on how to use or how to deploy the testbed.
The last threat that it will be mentioned is the status of the other frameworks, whether their evolution grows faster it could lead to a decay of the concerned software.

6.2 Description of configuration parameters

In order to analyze the effects related to different configuration of Network Slicing exploited at the RAN segment, several factors should be taken into account. In the following list, those parameters will be described.

1. Slice Configuration: one of the most important element that will be considered is the Resource Blocks distribution. Hence, how to assign the RGBs to each UE or group of UEs. The channel exploited in simulation environment allows a total bandwidth of 17/17.5 Mbps, that correspond to an amount of 13 RBG (12.5), numbered from 0 to 12. Hence, one of the goal of the following tests will be the management of these resources in order to achieve the best RTT performance.

2. Number of Users: a second parameter to be tuned is the number of users in the tested network, in order to analyzed whether the increasing number of them can influence the scheduling delay. The Moasic5G framework should support up to 127 UEs, anyway in this experiment will be treat only up to 50 UEs.

3. User typology: not only the total number of UEs which is present in the network is noteworthy, but also the percentage of mobiles belonging to different services can influence the results. Therefore, it will be study even the impact of varying the number of URLLC users, with respect to the eMBBs.

4. Scheduling algorithms: it will be analyzed the way different schedulers affect the RTT, accordingly with the emulated scenario.

5. CQI: even the quality of the channel can affect the results. It will be take into account also the behaviour of mobiles experiencing different channel conditions. This value in a real environment is reported to the BS from each UE periodically. In this testbed, it is simulated and can be manually set. It is represented by a number from 0 to 15, where 15 means the mobile is experiencing perfect channel conditions. The CQI parameter can be set to a fixed value, or can be randomly generated every TTI.

6. Traffic scenario: different configuration of Down Link traffic are emulated. Indeed, different typologies of data flows impact the schedulers performances.

Parameter	**Description**	**Admissible values**
Slice width	Associating the number of RGB dedicated to each slice.	[0-12] RGBs
Number of UEs	Mobile users present in the network simulation.	From 0 to 127
Service types	Typology of services required by the UEs	URLLC/eMBB
Algorithm	Scheduling algorithm implemented at the eNB. Can be different for every slice.	RR, PF, MT
CQI	Telnet endpoint simulating the channel status experienced by every UE. Can be a static value or variable along time.	From 0 to 15
Traffic scenario	Amount of data downloaded by UEs: allow to simulate different services.	Up to an aggregate of 17.5 Mbps

Table 6.1: Configuration parameters in a nutshell.

Hence, the first part of the analysis focuses on the study of Round Robin behaviour. It has been fixed the scheduling algorithm and the typologies of flows, connected to the users service type. The parameters which have been sequentially changed are the aggregate traffic in the network related to the slice space, the whole number of mobiles and the percentage of type of users.

Subsequently, the analysis focused on the comparison among the three different scheduling algorithms implemented in the Mosaic5G framework: Round Robin, Proportional Fair and Maximum Throughput.
In particular, the latter, when making decision, consider also the quality of the channel experienced by each user. Therefore, in order to analyze the behaviour of them, the network has been emulated generating random CQIs and the schedulers performances are observed when the two classes of interest, URLLC and eMBB, are present while the channel state is time-varying.
Lastly, pointing out the great response of Proportional Fair in this particular use case, it has been deeply studied its trend in more realistic situations. Indeed, considering the URLLC flows constant at a low data rate is a reasonable simplification for the simulation. However, a more accurate simulation of URLLC traffic is expecting an intermittently low/medium data rate flow. Therefore, it has been observed the behaviour of slicing in these situations.

As final step, some policies have been implemented to automatize the process of resource management. The study of the behaviour of the network in different traffic conditions, will help to understand which policies develop to optimize the resource management in analogue scenarios.

6.3 Impact of slicing with Round Robin

The initial part of this work deals with the analysis of a network handled by Round Robin in ideal conditions. Considering ideal conditions, it corresponds to assume the absence of Noise along the channel, which translates into users experiencing the maximum bandwidth available. Under this scenario, it will be investigated the behaviour of Round Robin algorithm when working with different configurations.

The principal case study includes two categories of users, URLLC and eMBB, which will be simulated with different traffic characteristics.
More in details, the URLLC users, which should represent preemptive messages, like alarms, are characterized by a small amount of data which have to be transmitted faster and with an high level of reliability. Therefore, it has been used a download application of 10 kbit/s for each user. On the other hand, the eMBB mobiles are characterized by an intense flow of data, which vary depending on the total number of users and on the limits of simulator. Indeed, the maximum aggregate admissible data rate is around 17.5 Mbit/s which represents a strong constraint for the tests.

Every simulation will be analyzed in terms of average Round Trip Time and Standard Deviation (Std Dev) for group of users. Further, to clearly understand

the behaviour, it will be computed the percentage variation $\Delta\%$ between the results with and without NS management: this value will represent how large is the gain of RTT or Std Dev whether positive, or the lack if negative.
In addiction, all the test performed in this section have been computed with simulations of 800 s, eliminating the 3% of higher and lower values in order to smooth the results and avoid uncontrolled peaks.

6.3.1 Analysis of slice occupancy

As first scenario, it will be considered a fixed number of users equal to 20, characterized as follow: the 25% belongs to URLLC services and the remaining 75% to eMBBs. With this setting, it has been investigated the differences on users performances when Network Slicing is applied or not. In details, in each NS configurations the whole channel is shared into two different and isolated slices, each of them handled by a scheduler following the Round Robin fashion. Both slices are deployed on the same eNB and share the same spectrum. For every configuration of Resource Blocks, the slice associated to eMBB mobiles, is tested when filled with different quantity of data.

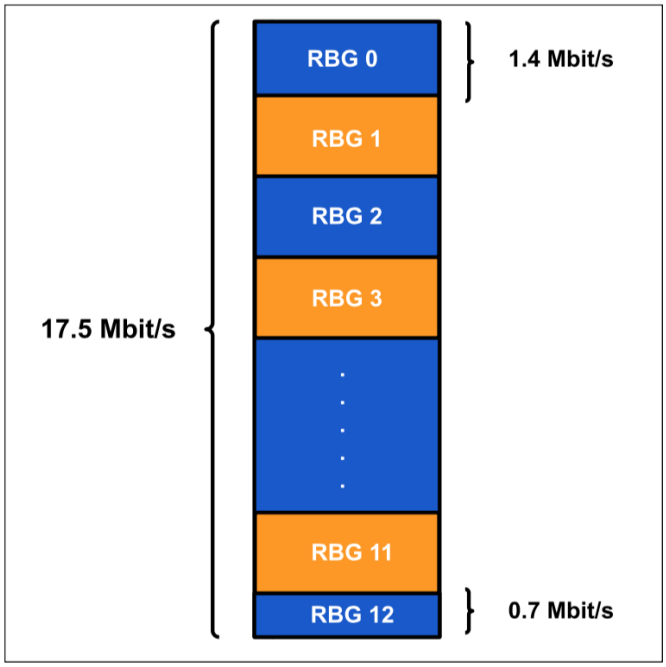

Figure 6.2: Down Link Channel Resource Blocks

Fig. 6.2 shows the Resource Group Blocks (RGBs) provided by the simulator. At the time of writing, Mosaic5G has not yet released other slicing modalities, unless the static one. This means that it is not possible to handle few bits of bandwidth, thus the resource distribution concerns only blocks of space, 1.4 Mbit/s large.
In this respect, the slices configurations chosen are:

- **NS configuration 1**:
  - Slice 1 "*eMBB*": blocks [0-9];
  - Slice 2 "*URLLC*": blocks [10-12];
- **NS configuration 2**:
  - Slice 1 "*eMBB*": blocks [0-10];
  - Slice 2 "*URLLC*": blocks [11-12];
- **NS configuration 3**:
  - Slice 1 "*eMBB*": blocks [0-11];
  - Slice 2 "*URLLC*": blocks [12-12];

Each NS configuration is tested with different level of occupancy of eMBB slice, therefore increasing the the throughput of each eMBB user.

NS configuration 1:

The first configuration of Network Slicing associates 10 RGBs to Slice 1, dedicated to eMBB users, corresponding to a channel bandwidth of 13.9 Mbit/s. While the last 3 blocks, from 10 to 12 RGBs, which support a bandwidth of 3.6 Mbit/s, have been dedicated to Slice 2, for URLLC mobiles.
The behaviour of the system is examined when the channel is filled with quantities of data equal to the 70%, 80% ,90% and 100% of "*eMBB*" Slice 1 width.

Occupancy	URLLC UEs			eMBB UEs		
	✗ NS	✓ NS	Δ%	✗ NS	✓ NS	Δ%
70 %	19.20 ms	19.08 ms	-0.63%	19.99 ms	23.34 ms	+16.76%
80 %	19.44 ms	19.03 ms	-2.11%	23.66 ms	31.98 ms	+35.16%
90 %	21.27 ms	19.00 ms	-10.67%	27.71 ms	34.95 ms	+26.13%
100 %	22.12 ms	19.1 ms	-13.65%	31.97 ms	37.37 ms	+16.89%

Table 6.2: Average Round Trip Time with NS configuration 1

Occupancy	URLLC UEs			eMBB UEs		
	✗ NS	✓ NS	Δ%	✗ NS	✓ NS	Δ%
70 %	2.17	2	-7.83%	2.13	3.4	+59.62%
80 %	3.8	2.08	-45.26%	8.02	2.75	- 65.70%
90 %	4.65	2.6	-44.09%	14.11	5.38	+61.87%
100 %	3.8	2.39	-37.11%	5.5	7.2	+30.90%

Table 6.3: Average Standard Deviation with NS configuration 1

From Tables 6.2 and 6.3 is pointed out that increasing the "*eMBB*" slice occupancy, which corresponds to a larger aggregate amount of traffic in the channel, the gain achieved from URLLC mobiles exploiting slicing is larger. Indeed, resource isolation allows the low-latency devices to experience always the same queuing delay, despite the growing eMBB throughput in Download. On the other hand, eMBB RTTs with NS increases as the traffic conditions get worsen.

NS configuration 2:

The second case analyzes the effect of varying the *eMBB* slice occupancy when the second NS configuration is chosen. Hence, 11 blocks have been assigned to the first slice *"eMBB"*, from 0 to 10, allowing a bandwidth of 15.3 Mbit/s. The last two blocks, 11 and 12 are associated to URLLC services, which can experience up to

2.2 Mbit/s . The experiment considers different levels of occupancy, such as 70%, 80% , 90% and 100% of 15.3 Mbit/s.
The following test, given the same URLLC aggregate throughput in DL, aims to investigate the consequences of reducing the resource dedicated to slice 2 (*"URLLC"*), while increasing the total amount of data received by eMBB mobiles .

Occupancy	**URLLC UEs**			**eMBB UEs**		
	✗ NS	**✓ NS**	Δ%	**✗ NS**	**✓ NS**	Δ%
70 %	20.85 ms	18.95 ms	-9.35%	24.02 ms	35.9 ms	+49.50%
80 %	20.70 ms	18.90 ms	-8.70%	24.60 ms	30.99 ms	+26.00%
90 %	21.56 ms	18.99 ms	-11.92%	27.55 ms	28.93ms	+05.01%
100 %	22.93 ms	18.62 ms	-17.58%	29.31 ms	72.33 ms	+146.8%

Table 6.4: Average Round Trip Time with NS configuration 2

Occupancy	**URLLC UEs**			**eMBB UEs**		
	✗ NS	**✓ NS**	Δ%	**✗ NS**	**✓ NS**	Δ%
70 %	3.60	2.48	-31.11%	5.87	3.27	- 44.29%
80 %	3.13	2.00	-36.10%	6.80	6.98	+02.65%
90 %	3.83	2.33	-39.16%	14.24	4.1	- 71.21%
100 %	4.25	2.46	-41.98%	6.69	14.47	+116.3%

Table 6.5: Average Standard Deviation with NS configuration 2

As it is shown from results in Tables 6.4 and 6.5, exploiting NS configuration 2, even if the resource allocated to URLLC have been reduced, the benefits of applying NS are still evident. Indeed, the RTT of low-latency mobiles is almost the same as the previous case, sign of that the space for this flows of data is still enough. Or better, the first configuration was leaving too resources to URLLC, which were wasted.
Comparing the RTTs of the two situations in absence of NS, in the second configuration result slightly higher. This is due to the fact that the traffic flowing in the channel is increased, leading to an increment in the queuing waiting time, too. However, as regard for NS scenarios, while URLLCs experience the same delay, eMBBs need to wait slightly more. That happens even if the amount of data presents in the network increases proportionally to the Slice 1 space. Reasonably, it can be a consequence of the larger quantity of packets: even if the proportion between channel space and traffic is the same as the previous case, the scheduler handles longer queues. Round Robin algorithm is completely fair in number of packet per queue, but unfair in terms of bit per time slots. This means that for variable packet size, before switching to the next queue the current packet transmission must finish. Therefore, larger packet will globally occupy the channel more. This is the reason way in NS case for broadband users, the higher the bit/s per UE, the higher the RTT.
Therefore, this resource management allows to achieve the same URLLC latency gained in NS configuration 1, while hosting larger quantities of bits.

NS configuration 3:

To conclude, the last resources configuration, which exploits as less as possible blocks for *URLLC* slice, such as the 12th block, that support a throughput of 0.7 Mbit/s. The remaining resources, 16.8 Mbit/s, have been associated to the *eMBB* slice. Following the previous tracks, the third NS configuration is tested with multiple occupancy levels of *eMBB* slice.

Occupancy	URLLC UEs			eMBB UEs		
	✗ NS	✓ NS	Δ%	✗ NS	✓ NS	Δ%
70 %	21.37 ms	23.14 ms	+8.28%	26.6 ms	28.27 ms	+06.28%
80 %	21.43 ms	23.7 ms	+10.59%	25.63 ms	28.35 ms	+10.61%
90 %	23.36 ms	23.89 ms	+2.27%	35.26 ms	28.17 ms	- 20.21%
100 %	24.00 ms	23.80 ms	-0.83%	43.50 ms	1078 ms	+2378%

Table 6.6: Average Round Trip Time with NS configuration 3

Occupancy	URLLC UEs			eMBB UEs		
	✗ NS	✓ NS	Δ%	✗ NS	✓ NS	Δ%
70 %	4.34	2.87	-33.87%	7.60	10.59	+39.34%
80 %	4.3	3.3	-23.26%	7.9	8.5	+07.59%
90 %	4.8	3.58	-25.42%	15.67	5.5	- 64.90%
100 %	4.3	3.6	-16.28%	32.84	686	- 1988%

Table 6.7: Average Standard Deviation with NS configuration 3

As it noticeable from the Tab. 6.6 and 6.7, considering the smallest amount of resources assigned to URLLC users, applying slicing it does not lead to any improvement. Indeed, the slice considered is too small to allow a better performance. However, the only positive result is found in the last case (occupancy of 100%) because the channel conditions without NS have worsened enough, so it becomes possible to appreciate the benefits of NS. Therefore, if it would be possible to fill the channel with more data, the gain would be even higher.

To sum up, these analysis have been conducted to prove the proper operation of the testbed and to highlight the effectiveness of the network slicing approach. Concluding, it is pointed out that the second NS configuration is the more suitable for the traffic received by low-latency users in this scenario, leading to the highest gain without waste. Therefore, with reference to NS configuration 2 and an *eMBB* slice filled for the 90%, the behaviour of the RTT along time is shown in Fig.6.3

and 6.4 for the case without and with NS.

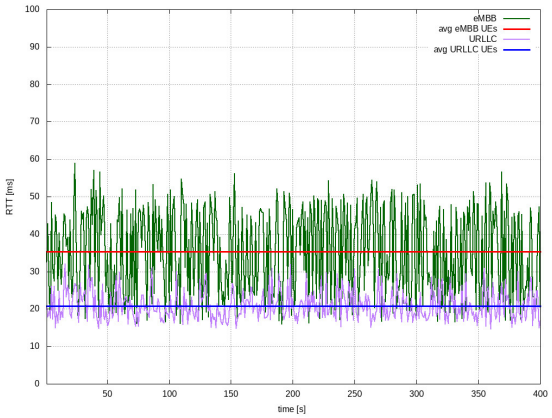

Figure 6.3: Without NS

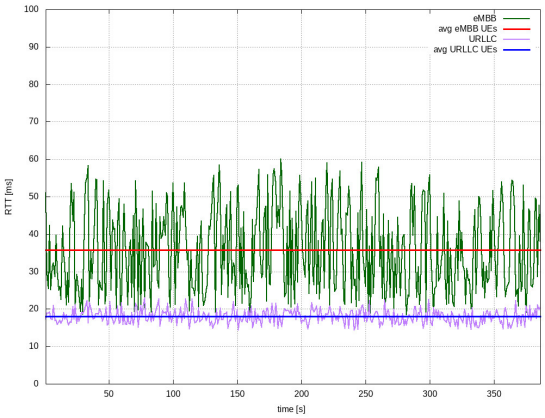

Figure 6.4: With NS config. 2

The benefits introduced by NS approach are evident from the graphs above: when applying resource separation for traffic services, the average RTT of low-latency mobiles is reduced of some milliseconds, as well as its variability. Indeed, the values fluctuation range between a narrower interval of values. On the other hand, in order to support the same large traffic, the RTT of other users is penalized and increases.

In this case the scheduler handles all the UEs almost at the same way, giving the same chance to each one. This results in a smooth RTT for all of them. However, the eMBBs experience an higher RTT (almost 10 ms) most of the time. This is due to the fact that they are receiving larger packets than URLLC mobiles, therefore they occupy the channel for a longer period and ICMP packets need to wait more time.
Moreover, even the standard deviation is reduced. This is a sign of that every URLLC user in average wait the same amount of time in the scheduling queue and the variability is more controlled. Differently from the case with absence of resource management, where the behaviour was more fluctuating. As an evidence of that, it is possible to notice in Fig. 6.3 that sometimes the medium RTT of URLLC users reaches, or even overcomes, 30 ms. While exploiting the FlexRAN controller functionalities, the overall trend it results much more controlled and stable.

Furthermore, it is compared the behaviour of slicing when exploited carefully or

not.
As regard for the first pair of figures (Fig. 6.3 and Fig. 6.4), they represent the trend in absence of resource isolation on the left side, and benefits of applying a clever slices management on the right one. There is an evidence of how the RTT for URLLC decreases when separating the resources.

Whereas, in the next plots, it has been demonstrated a naughty usage of slicing, such as assigning the RGBs in an unbalanced way.

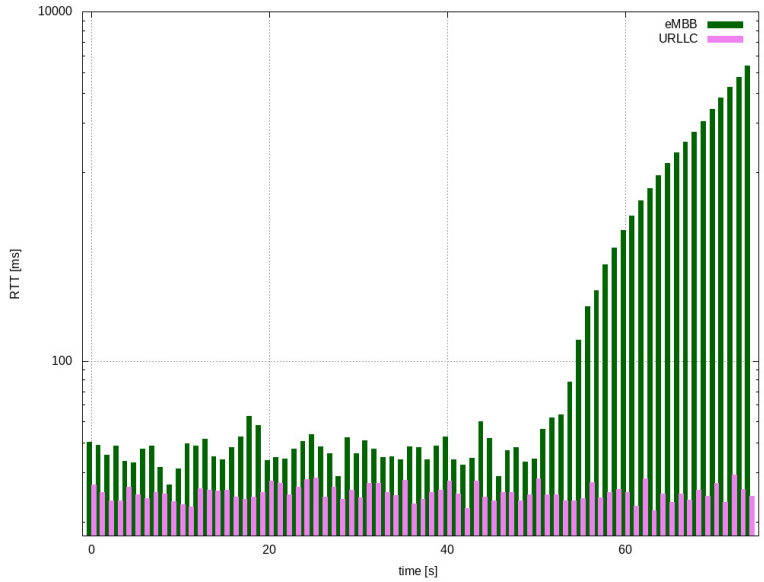

Figure 6.5: Slice overfilled.

Hence, if increasing too much the resource blocks assigned to URLLC users, what it turned out is that the advantage in RTT does not increase consequently. Indeed, since the amount of data exchanged between low-latency users is just 50 kbit/s in total, assigning an excessive quantity of bandwidth becomes useless. While, for the other group of devices, it becomes even more complicated having a good RTT with a channel constantly overfilled.
In particular, while the URLLC group does not report any particular advantage, as for the eMBB group, reducing too much the eMBB slice, implies a congestion situation on that part of the channel. As it is possible to state from figure 6.5, the RTT drastically increases as a sing of engorgement.

To conclude, through these experiments it has been possible to verify the correct functioning of the testbed and the effectiveness of NS approach in reducing the

URLLC latency. Furthermore, the overall target of these assessments does not deal with finding the absolute latency profit but, since the simulations involve just a few users, with understanding how and when NS impacts latency.

6.3.2 Analysis with different number of UEs

After investigating the behavior of different levels of slice occupancy, relying on previous results, it will be selected the NS configuration 2 and observed the trend with different numbers of Users.
It will be considered the eMBB slice occupied for the 90%, leading to a channel bandwidth of 13.77 Mbit/s. While, as for URLLC, the resources associated to them are 2.2 Mbit/s.
In addiction, the percentage of low-latency UEs is constant, equal to the 25% of the total amount of mobiles and the data rate downloaded by each URLLC mobile (10 kbit/s). Last but not least, the scheduler is always the Round Robin.

The following tables represent the Round Trip Time and the Standard Deviation measured increasing the global number of devices present in the network.

Users	**URLLC UEs**			**eMBB UEs**		
	✗ NS	**✓ NS**	**Δ%**	**✗ NS**	**✓ NS**	**Δ%**
8	18.19 ms	16.50 ms	-9.29%	45.89 ms	41.52 ms	- 9.52%
12	21.35 ms	18.46 ms	-13.54%	42.10 ms	46.20 ms	+9.74%
20	23.20 ms	19.19 ms	-17.28%	31.70 ms	34.12 ms	+7.63%
32	38.64 ms	31.54 ms	-18.37%	28.75 ms	45.97 ms	+59.89%
40	36.65 ms	31.66 ms	-13.62%	31.21 ms	43.24 ms	+38.55%
52	51.25 ms	46.48 ms	-9.30%	39.90 ms	54.74 ms	+37.10%
60	52.62 ms	48.07 ms	-8.65%	45.60 ms	69.70 ms	+52.85%

Table 6.8: Average Round Trip Time with different numbers of UEs.

Users	URLLC UEs			eMBB UEs		
	✗ NS	✓ NS	Δ%	✗ NS	✓ NS	Δ%
8	2.46	9.26	+276.4%	15.6	3.08	-80.26%
12	3.7	2.8	-24.32%	6.7	42.9	+540.3%
20	4.6	2.4	-47.83%	8.2	9.57	+16.71%
32	5.4	2.47	-54.26%	11.02	11.7	+6.17%
40	4.86	2.12	-56.38%	3.86	5.6	+45.08%
52	9.47	6.3	-33.47%	3.64	4.6	+26.37%
60	10.8	8.4	-22.22%	4.65	8.04	+72.90%

Table 6.9: Average Standard Deviation with different numbers of UEs.

Analyzing the results collected in Tab. 6.8, it is pointed out that the gain for URLLC increases with the number of users up to a maximum, and then decreases again. That is due to the fact that an increment of the mobiles, can lead to an increment in the advantages of exploiting NS, but up to a certain point. When the number of users raises too much, the network becomes more congested and the gain is less effective.
However, this result is also influenced by the RR scheduler implementation, which serves the user in sequence so, the higher the number of users, the higher the waiting time for each of them. Therefore, when NS is applied low-latency users need to wait for smaller amounts of time since are isolated from eMBB users.

6.3.3 Analysis with different percentage of URLLC users

In the following scenario it is evaluated the network behaviour at the variation of the number of URLLC users. Therefore, it has been considered a pool of 20 users in total, and observed how the performances changes when the percentage of URLLC UEs is different.
The scheduling algorithm considered is always Round Robin. And for this traffic configuration the NS configuration exploited is still the second, with a global amount of data in the channel equal to the 90% of the eMBB slice width, therefore 13.77 Mbit/s aggregate for this users category. Low-latency users receives always 10 kbit/s each, and what happens for them is reported in the following tables.

UEs	URLLC UEs			eMBB UEs		
	✗ NS	✓ NS	Δ%	✗ NS	✓ NS	Δ%
5%	22.46 ms	18.57 ms	-17.32%	32.16 ms	33.02 ms	+2.67%
20%	20.85 ms	19.15 ms	-8.15%	25.17 ms	38.29 ms	+52.13%
50%	22.02 ms	20.44 ms	-7.18%	33.53 ms	34.66 ms	+3.37%
70%	21.55 ms	20.57 ms	+4.55%	28.84 ms	29.43 ms	+2.05%
80%	20.19 ms	21.22 ms	+5.10%	42.18 ms	40.28 ms	+4.5%
90%	18.37 ms	21.34 ms	+16.17%	29.52 ms	51.14 ms	+73.24%

Table 6.10: Average Round Trip Time varying URLLC UEs

UEs	URLLC UEs			eMBB UEs		
	✗ NS	✓ NS	Δ%	✗ NS	✓ NS	Δ%
5%	6.79	4.60	-32.25%	3.94	4.30	+9.14 %
20%	3.83	2.90	-24.28%	5.10	6.70	+31.37%
50%	3.39	1.64	-51.62%	5.33	4.38	-17.82%
70%	6.55	2.30	-64.89%	19.67	9.00	-54.25%
80%	2.68	3.27	+22.01%	20.24	17.75	-12.30%
90%	1.58	3.82	+141.8%	10.56	23.19	+119.6%

Table 6.11: Average Standard Deviation varying URLLC UEs

Therefore, with the increasing of the number of URLLC mobiles in the network, even if they occupy the channel with a very small amount of data, the advantage using slicing is reduced, if not even leading to a disadvantage for them.
Indeed, the resources dedicated for low-latency users are always the same, even when the their percentage in the channel is higher. Maybe in this cases, it would be better to increase the amount of RGBs associated to URLLC slice.

6.4 Impact of different scheduling algorithms

So far, is has been considered only the Round Robin scheduler, which is the most fair in term of chances to every user, but it does not consider the instantaneous throughput or the quality of the channel. Therefore, it will be analyzed the behaviour of other algorithms, such as Proportional Fair and Maximum Throughput,

which schedule the mobiles considering also the data rate and the CQI reported by every user.

Up to this point, the channel conditions have not been considered. Anyway, in a real environment Noise is present along the channel, worsening the transmissions. In order to take into account this factor, every UE sends to the eNodeB an indicator, raging from 0 to 15, which indicates the quality of the channel based on the Signal-To-Noise Ratio (SNR). This symbol is the Channel Quality Indicator (CQI), which is a measurement of the channel status reported by every mobile user based on its past experience, changes along time according to the channel state.

The presence of Noise in a channel it depends on multiple factors, and it becomes interesting and challenging also study how to improve digital communications for poor quality channels.
In cellular networks, the users with poor channel quality usually experience poor QoS. To satisfy QoS requirements of all users, BSs have to adjust their transmitting power to satisfy users with the poorest QoSs.

The UE is realized to report periodically CQI values, measured from the physical layers. However, the testbed has been deployed in a L2 simulation environment thus, bypassing the physical channel. Hence, in order to analyze this scenario Mosaic5G framework offers a simulation of the channel state, such as a variable value that affects the channel sensing. Especially, in L2 simulator this control message can be set through a Telnet endpoint following two modalities:

- Mode 1: select a time-fixed CQI for the UE.
- Mode 2: select a random time-varying CQI, which changes every TTI.

Aiming to evaluate the impact of different scheduling algorithms in a scenario where broadband services need to coexist with mission-critical ones, and most of the devices move in space, changing the perceiving bandwidth, the next experiments have been conducted.

The following simulations consider an environment where each user experiences a random CQI (Mode 2), generated casually between 0 and 15 every TTI. The value 15 means the channel is perfect, noiseless. Therefore the bandwidth is the maximum achievable (17.5 Mbit/s). The lower the CQI value, the narrower the channel.

Firstly, it has been investigated the differences between the schedulers in terms of throughput achieved and losses in a traditional situation in absence of NS. The

scenario is populated by 20 devices: the 25 % behaves as low-latency services, receiving flows of 10 kbit/s, and the remaining 75% collects broadband data at 0.8 Mbit/s each, for a total of 12 Mbit/s. The results in terms of aggregate throughput and percentage of lost packets for each algorithms are the following.

Alg.	**URLLC UEs**		**eMBB UEs**	
	Throughput	Loss %	Throughput	Loss %
RR	0.01 Mbps	~ 0	9.012 Mbps	13.60 %
PF	0.01 Mbps	~ 0	11.77 Mbps	1.69%
MT	0.01 Mbps	~ 0	11.96 Mbps	0.45%

Table 6.12: Throughput and losses comparison among schedulers.

As it shows Tab.6.12, while for URLLC UEs the trend is almost the same, as regard broadband services each algorithms presents some differences.
As expected from their implementation, RR is the one that reaches the smaller throughput in mean, leading to a large percentage of packets lost, too. Indeed, RR algorithm does not take into account the CQIs, serving users in order. On the other hand, MT which schedules CQIs in order of decreasing, is able to reach almost the maximum bit rate, collecting the lower number of packets dropped. Finally, PF is implemented to find a balance between channel state and data rate, thus it experiences a smaller throughput than MT, but still very high.

The previous scenario is then exploited to investigate the behaviour of the schedulers in terms of RTT, when NS is applied.
This time the total amount of data flowing in the channel is reduced, in order to avoid crushing the testbed when NS is considered. Therefore, it is found that the combination of bit-per-second sustainable for eMBB mobiles is 0.51 Mbit/s each user, for a total of 7.65 Mbit/s. This eMBB aggregate throughput represents the 55% of the *"eMBB"* slice presents in NS configuration 1, described in Sec. 6.3.1. The bit-rate for URLLC UEs, instead, remains the same as before.

Under this scenario it will be presented a comparison of the behaviour and the characteristics of each algorithm, when URLLC and eMBB services occur together.

Alg.	**URLLC UEs**			**eMBB UEs**		
	✗ NS	**✓ NS**	**Δ%**	**✗ NS**	**✓ NS**	**Δ%**
RR	24.70 ms	18.96 ms	-23.24%	48.57 ms	2360 ms	+ 4758%
PF	18.76 ms	19.13 ms	+1.97%	33.00 ms	49.95 ms	+51.36%
MT	32.20 ms	19.09 ms	-40.71%	31.36 ms	49.71 ms	+58.51%

Table 6.13: Schedulers comparison: average Round Trip Time

Alg.	**URLLC UEs**			**eMBB UEs**		
	✗ NS	**✓ NS**	**Δ%**	**✗ NS**	**✓ NS**	**Δ%**
RR	3.99	1.91	-52.13%	18.78	910	+4745%
PF	2.22	2.3	+3.60 %	7.65	11.5	+50.33%
MT	11.44	2.37	-79.28%	7.7	10.67	+38.57%

Table 6.14: Schedulers comparison: average Standard Deviation

Comparing the performances of the three schedulers, it turns out that Maximum Throughput is the one that reflects the higher percentage of gain through NS, while Proportional Fair achieves the best RTT for mission-critical services also sharing the channel with eMBB users.
In fact, MT schedules as first the mobiles with the highest CQI without considering the data-rate. This reflects into a very similar RTT for URLLC and eMBB when no slicing configuration has been selected, indeed MT algorithm is thought to maximize the throughput. Therefore, this time the improvement for this use case is strongly evident. With the same network configuration and the same Down Link traffic, the latency experienced by the users is shown in the figure 6.6 below.
On the other hand, PF aims to find a balance between best channel conditions and lower throughput, so that even if a UE has a poor CQI it will get resource. Actually, the trend of PF, since in this scenario all the UEs get a random CQI, rather than avoid starvation of UEs with worst channel conditions, results prioritize URLLC even without NS. This might be justified by the fact that associating a scheduling priority to each jobs inversely proportional to its past resource consumption, helps to ensure a minimum level of service in disadvantageous conditions but consider prioritized also UEs with lower throughput. Therefore, applying slicing in this context does not lead to any advantage: the RTT measured on average also without slicing is already almost minimum.
Anyway, to scheduling URLLC users before, the eMBB devices wait more time. Indeed, their RTT is slightly higher than with MT.

Lastly, the behaviour of RR is the same analyzed in the previous section: indeed the CQI it is not considered by this algorithm.

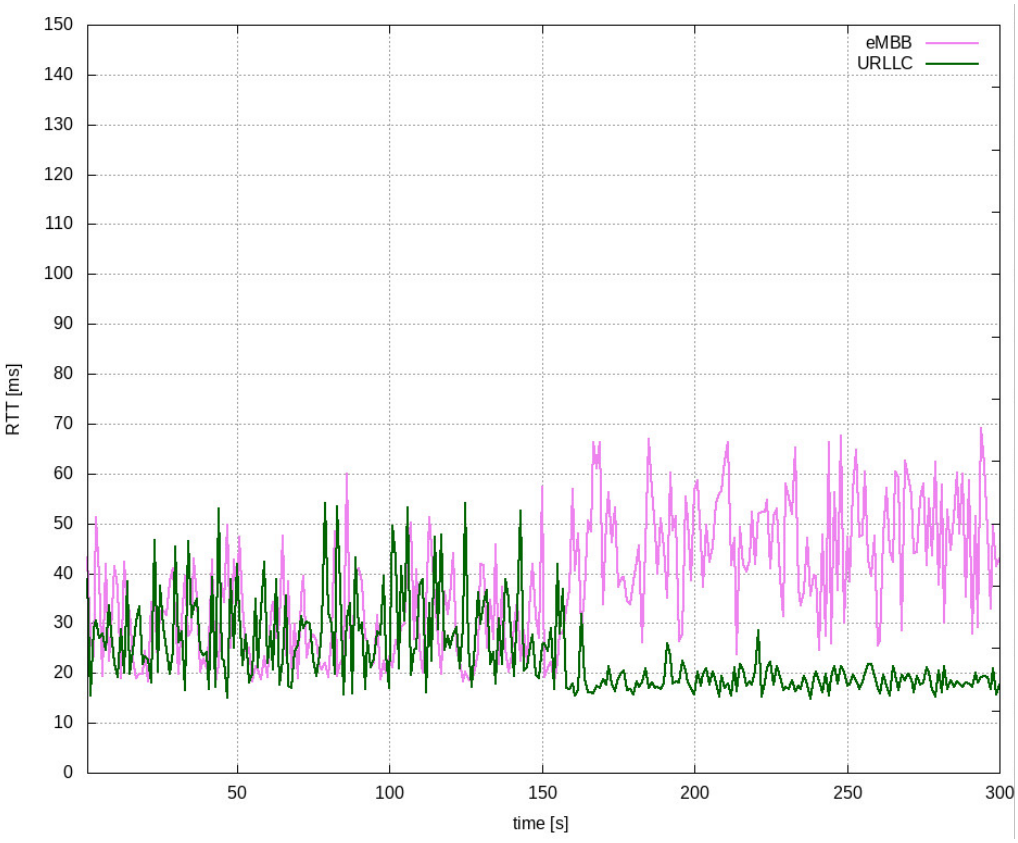

Figure 6.6: RTT trend with MT: before and after slicing application.

At the beginning of the plot in Fig. 6.6, there is an evidence of how the average RTT of URLLC mobiles is really high, reaching 40/50 ms. Indeed, MT algorithm queues as first the users with the higher CQI in order to transfer as much as possible data. Therefore the queue of URLLC UEs, even if they receive a very small amount of data, is very low. However, at time 150s, a slicing configuration has been applied to the scenario. Since the goal is to preemptive the low-latency mobiles, improving their experience, a slot of resource blocks has been assigned exclusively to them. It results that one of the best configuration that reduces the URLLC delay but also does not penalize too much the other mobiles is assigning 3 RGBs to low-latency slice and the remaining 10 blocks to high-traffic users. Starting from the moment in which the proper slice configuration has been enforced, the average RTT for

URLLC mobiles strongly decreases. Furthermore, the trend of these users becomes constant and no more unexpected peaks are observed.

6.5 Particular case: URLLC UE receives intermittently data

Pointing out the great response of Proportional Fair into limiting the latency for URLLC users highlighted in the previous cases, it has been considered interesting studying further its trend, in more realistic situations.
Up to now, mission-critical services have been simulated with constant flows at low data rate. This was a reasonable simplification for the initial step of the analysis. However, a more accurate simulation of URLLC traffic is expecting an intermittently low/medium data rate flow. Therefore, it has been observed the behaviour of slicing with PF in these situations.

In this context, the following scenario includes 20 UEs, characterized by the 25% acting as URLLC and the remaining 75% of mobiles as eMBBs. This time while eMBB users are considered downloading at constant data rate (12M total: 0.8 M each UE) (as higher as possible in this channel simulator), the low-latency devices receive a constant low traffic of 10 kbit/s with some peaks of 2 Mbit/s.

In the first scenario the CQI has been considered constant for all the users and maximum. Hence, in order to reduce the randomness, it has been considered an ideal situation where all the users are static and experience the best channel conditions.
In the following, it will be shown the behaviour of Proportional Fair scheduler in absence of resource management when URLLC flows are more variable.

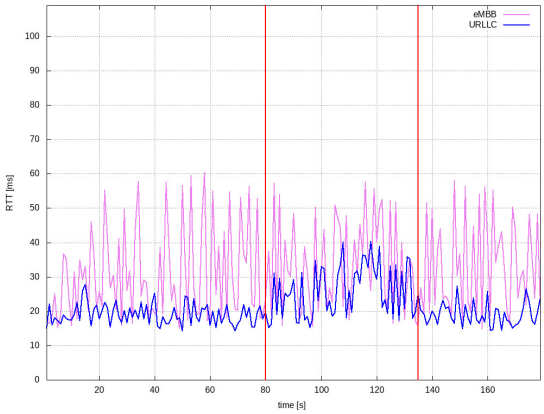

Figure 6.7: Common shared channel.

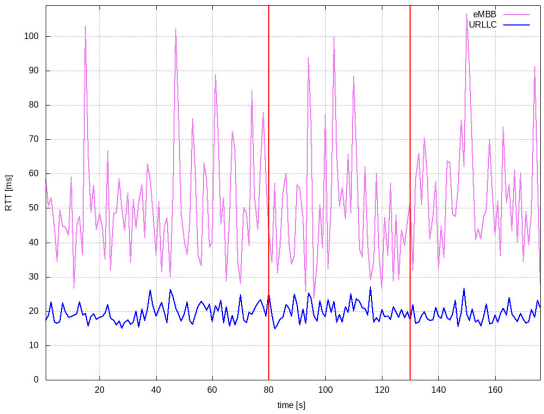

Figure 6.8: Slice isolation.

It is clearly visible from the average Round Trip Time plot that until URLLC UEs download a constant-low traffic, the scheduler performs very well even if all the resource are commonly accessed. However, as soon as one low-latency mobile receives a larger flows of data, its latency is strongly penalized, affecting the mean RTT.
The reason behind this penalization reflects the algorithm implementation that favours the mobiles with the highest instantaneous achievable data rate relative to its past average data rate. That in other words, when the channel quality is equal for all, means giving priority to who has the lower throughput.

Therefore, the test has been repeated considering exactly the same scenario, but applying resource isolation. The slices configured involve 9 RGBs for eMBB and the last 4 RGBs for URLLC. That in terms of bandwidth means a maximum space of 12.6 Mbit/s for the first typology and 4.9 Mbit/s for the other.
Running again the simulation with the same conditions, the eMBB Round Trip Time suffers the resource reduction, increasing of a ten of milliseconds. While the URLLC users, thanks to the slice isolation, experience a faster connection, reducing the scheduler delay.
In Figure 6.8 it is presented the RTT graph when slice management is applied. The red vertical lines delimit the time of arrival of the URLLC peak of data.

To conclude, owing to resource isolation it is possible to change the behaviour of the network, enhancing or worsening the delay experienced by the users. Moreover, a clever analysis of the situations, can help to decide the best structure to apply in order to achieve the target desired.
As regard, for instance, for this scenario the slices configuration chosen is the more suitable to limit the RTT increment for low-latency devices, while avoiding the

congestion of eMBB, which are just slightly penalized.

The same situation has been simulated but with random CQIs for all the users. In addiction, since with lower CQI the channel bandwidth is smaller, it has been reduced the aggregate throughput eMBB. The results show the same behaviour as the previous case.

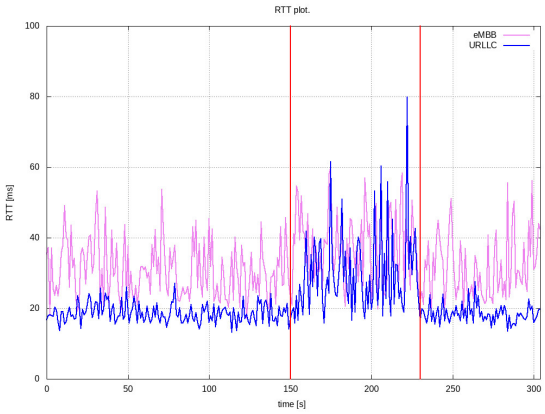

Figure 6.9: CQI case: without NS

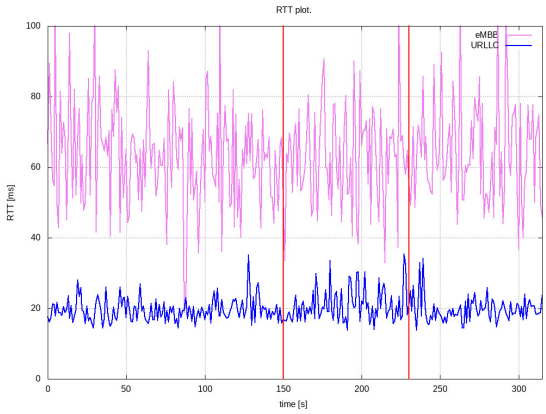

Figure 6.10: CQI case: with NS

6.6 Policy implementation

Once analyzed and understood the behaviour of slicing in different scenario, the idea is to automatize the process in a clever manner.
Therefore, it have been implemented some policies, based on the knowledge of RTT and throughput.

Firstly, it has been developed a program based on the average RTT per group of users, re-assigning the resources associated on a time-based threshold. The policy worked but as soon as an high delay was detected, the congestion situation was already happening. Therefore, the next idea has been to prevent a congestion, controlling the resource assignment through the throughput experienced at each user interface, and as soon as the aggregate throughput of a group of users exceeds the threshold or decreases too much under that, blocks of resources are added or removed. This is done with the target of handling the channel space in a clever manner, adapting the resources assigned to each user on its throughput.
Going into details, the throughput at each user interface has been measured through a tool named TShark.
TShark is a network protocol analyzer that lets you capture packet data from a live network [22]. Thanks to TShark it has been possible to capture all the packets arriving during a time window and compute the instantaneous throughput.
The protocol analyzer captures on each user interface, saving the frame number and the time of arrival, specified by the options "-e ". To reading easier the information, some filters are set: such as the protocol type UDP, the source address (the server) and the packets length (1024 Bytes).
From the data collected by this capture, it has been possible to identify the throughput of each user as a measure of the packets arrived in a window of time.

Therefore, the policy is based on the throughput measured every 1 s, which is compared with some thresholds verifying if the value still belongs to the expected interval. The intervals correspond more or less to the RBGs dimension. Whether it satisfies the expectations, everything is working well and nothing will be changed. As soon as one of the thresholds is overcame, a command for resource management is set.
Hence, if the throughput exceeds the higher threshold it means that a user is receiving a larger quantity of data, which are not fitting anymore in the current slice. On the other hand, whether the bandwidth is under the lower threshold, the user is exploiting a portion of channel too large, wasting it.
Therefore, according to this idea, to avoid wasting of bandwidth or allowing an higher throughput whether necessary, the policy adjust the RGBs of the slice, accordingly. This makes the resource management more efficient, trying to avoid

congestion or improving the communications.

To demonstrate what explained so far, it will be show a typical case, such as an incoming call. Considering a scenario with two mobiles, one reserved to URLLC traffic, and the other representing the eMBB, the first will be associated to a separate slice, with a dedicated and fixed amount of resources, equal to two RGBs. The URLLC user will receive a constantly low data rate flow, of 10 kbit/s. Since this represents the low latency and high reliability traffic, it is exploited an isolated part of the channel dedicated exclusively to it, to avoid conflict with high-demanding users.
The eMBB mobile, instead, has been designed downloading at a medium/high data-rate, like 0.5 Mbit/s, with an higher peak of 4 Mbit/s in the middle of the transmission. This kind of flow could represent the incoming call that the users is receiving.

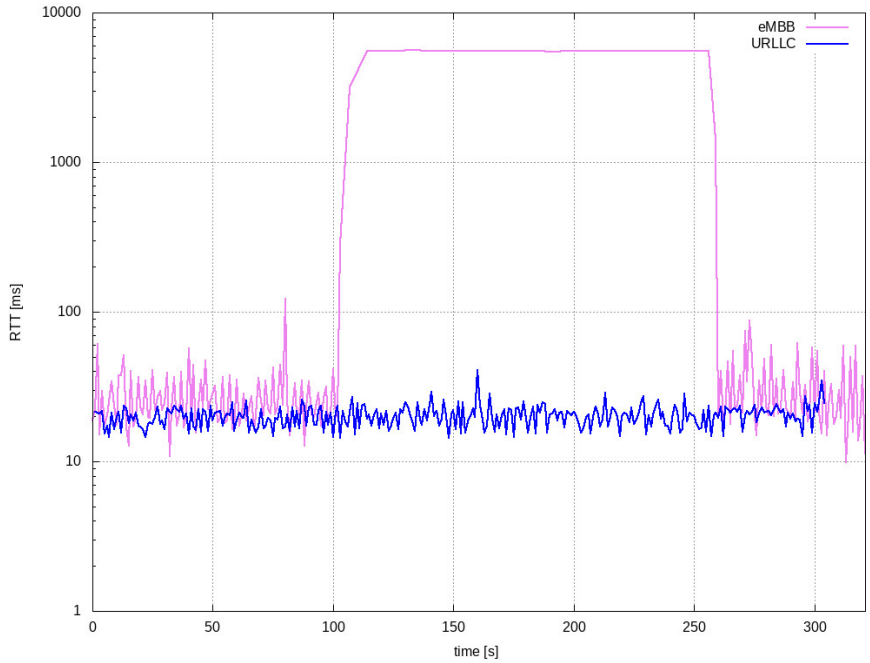

Figure 6.11: RTT in absence of policy.

As it is possible to see from the first graph (Fig. 6.11), if the resource associated does not change when the call arrives, the RTT drastically increases, sign of intense congestion.
Moreover, the throughput cannot reach the value desired, since the slice associated to eMBB is too small. Hence, the maximum achievable bandwidth is 1.4 Mbit/s,

namely 1 RGBs. This means that most of the incoming packets will be dropped and retransmitted, creating congestion along the channel. The evident drawback is the performance worsening.

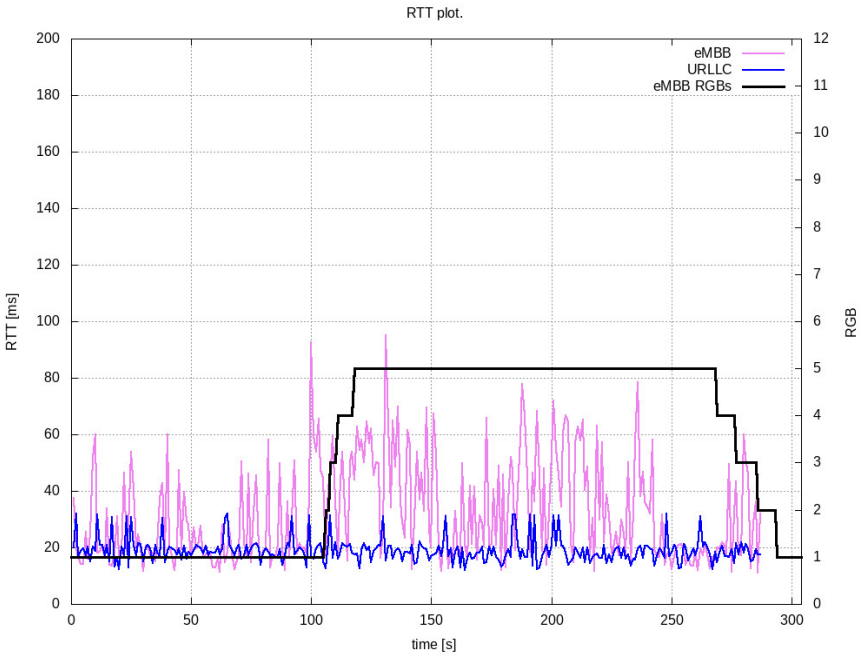

Figure 6.12: RTT when policy is applied.

In order to improve this misbehaviour, the same situation has been tested again but with the policy activated.
What is observable from the plot (Fig. 6.12) , is that the number of RGBs associated to the eMBB slice, varying according to the incoming traffic, increasing as son as the call arrives and decreasing when it ends. The RTT this time is much more controlled, and congestion is avoided.
As for the URLLC user, its RTT has not been affected by the increasing traffic, since it exploits another isolated channel.
This occasion, even the throughput experiences an improvement. Since the space associated is enough to accept all the incoming data, the data rate increases as much as possible.

Chapter 7

Conclusions

Future mobile networks will be required to sustain even more heterogeneous requirements, in terms of throughput, latency, reliability, availability, as well as operational requirements such as energy-efficiency and cost-efficiency. This condition arises from an increasing diversity of services carried by the mobile network, expanding the areas of interest to Industry 4.0, vehicular communication, or smart grid. The cornerstones to support these directions are flexibility and programmability, which are the main principles that allow to support such a network increasing density over a limited medium shared among multiple tenants, as well as satisfying different stringent requirements. A key enabler to address this fast and non homogeneous growth, it will be clearly Network Slicing, that thanks to its functionalities makes it possible to share a common and limited channel in a clever manner, assigning or releasing different amount of resources on demand. However, despite of the huge literature presents on that, the practical solution implementations are scarce, especially as regard for non commercial world.

Therefore, this work aims to highlight the importance of open-source solutions for NS purposes, which allows third-parties to deploy testbed for research purposes. Some benefits offered by these solutions are, first of all, the cost-effectiveness but also the practical feasibility of deployment. Moreover, making available software for implement mobile networks testbed which support almost all the features of a real network, it is very useful to speed up the enhancements and innovations in telecommunication world.

Furthermore, this thesis examines the capabilities of an open-source framework for implementing a mobile network supporting end-to-end NS, focusing on the resource management at RAN side. Slicing the RAN is a less mature and challenging to implement in practise, probably due to the shared nature of wireless resources, and it includes various Radio Access Technology (RAT) parameter configurations,

such as time and frequency resources, frame size, etc. Hence, this complicates the project realization, forcing to simplify the simulations. The goal deals with the investigation of how change the users experience when exploiting a SD-RAN architecture. In particular, the final target seeks to analyze the behaviour of the users when the network is required to support two different use case, URLLC and eMBB, characterized by extremely different requirements in terms of throughput and latency. The need of sustain these kind of scenario will be even more frequent in the future, just thinking about vehicular network or monitoring systems where critical information have to coexist with control ones.

The possibility of relying on a testing platform allows to understand in advance how the network behaves under some scenarios and facilitate the deployment of policies for resource management, maximizing the efficiency.
As the experimental results show, there are a long variety of factors that influence the network behaviour. A good knowledge of how they affect the performances can be helpful to predict the repercussions when something changes, or new services should be supported.

The future of mobile networks deals with customization, offering services even more tailored on tenants requirements, and Network Slicing leads to a flexible and on demand resource management, which allows to assign the physical resources to the services that require more or less bandwidth on the fly. As confirmed by practical results, Network Slicing affords most of the time a latency reduction for URLLC services, improving their experience and getting closer to the theoretical expectations.
Thanks to the knowledge gained with experience in simulation phase, in the final part of the work it has been possible to deploy and test a policy, which helps to automatize the process of resource distribution based on services needs. This permits to improve the users experience, as well as afford to different use cases to be supported simultaneously.
Moreover, it is highlighted that all the experiments conducted during this thesis are related to slicing at the eNB, therefore the gain in milliseconds obtained through these tests might be larger whether considering NS even at the core.

To conclude, the open-source world is fundamental for researchers but finding feasible platforms implementation for NS purposed is still an open problem. Even Mosaic5G, which appeared to be the more mature and complete project, it turned out to be limited under different aspects. Nevertheless, it has been possible to deploy a working testbed, and despite of the implementation constraints, several situations have been successfully analyzed. The SD-RAN platform achieved through FlexRAN, is a powerful tool with a rising future ahead. At the moment of

writing, the features actually available with this platform are still limited, but it is expected to be updated in the near future, merging new scheduling algorithms or dynamic slice management.

7.1 Future works

Despite the Layer 2 simulator limitations, Mosaic5G software platform results to be very attractive whether considering the trade-off between cost of deployment and features.
For its interesting applications, the testbed could be further exploited to analyse different use cases, including mMTC services in simulations, or to deploy more intelligent policies, creating algorithms able to deeper analyze the network status or to use information reported by the UEs, and re-distribute the physical resources choosing the best trade-off, or favouring a class of service with respect to another. Moreover, Mosaic5G platform could be deployed to extend the isolation to the CN, implementing slices E2E, including ll-MEC framework which allows the decoupling of CP and UP functionalities.

Bibliography

[1] Leonardo Bonati, Michele Polese, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. «Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead». In: *Computer Networks* 182 (2020), p. 107516 (cit. on pp. 5, 15).

[2] Ali Esmaeily, Katina Kralevska, and Danilo Gligoroski. «A Cloud-based SDN/NFV Testbed for End-to-End Network Slicing in 4G/5G». In: (Apr. 2020) (cit. on pp. 7, 14, 18, 19).

[3] Wanqing Guan, Xiangming Wen, Luhan Wang, Zhaoming Lu, and Yidi Shen. «A Service-Oriented Deployment Policy of End-to-End Network Slicing Based on Complex Network Theory». In: *IEEE Access* PP (Apr. 2018), pp. 1–1. DOI: 10.1109/ACCESS.2018.2822398 (cit. on p. 7).

[4] Mourice O. Ojijo and Olabisi E. Falowo. «A Survey on Slice Admission Control Strategies and Optimization Schemes in 5G Network». In: *IEEE Access* 8 (2020), pp. 14977–14990. DOI: 10.1109/ACCESS.2020.2967626 (cit. on p. 8).

[5] P. Rost et al. «Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks». In: (2017). DOI: 10.48550/ARXIV.1704.02129. URL: https://arxiv.org/abs/1704.02129 (cit. on p. 9).

[6] Gines Garcia-Aviles, Marco Gramaglia, Pablo Serrano, and Albert Banchs. «POSENS: A Practical Open Source Solution for End-to-End Network Slicing». In: *IEEE Wireless Communications* 25.5 (2018), pp. 30–37. DOI: 10.1109/MWC.2018.1800050 (cit. on pp. 9, 10, 19, 20).

[7] Dinesh Tamang, Sergio Martiradonna, Andrea Abrardo, Gianluca Mandó, Gabriele Roncella, and Gennaro Boggia. «Architecting 5G RAN slicing for location aware vehicle to infrastructure communications: The Autonomous Tram use case». In: *Computer Networks* 200 (2021), p. 108501 (cit. on p. 10).

[8] Ali Esmaeily and Katina Kralevska. «Small-Scale 5G Testbeds for Network Slicing Deployment: A Systematic Review». In: *Wireless Communications and Mobile Computing* 2021 (May 2021), pp. 1–26. DOI: 10.1155/2021/6655216 (cit. on pp. 14, 15).

[9] *Mosaic5G*. URL: https://mosaic5g.io (cit. on pp. 16, 17).

[10] Xenofon Foukas, Navid Nikaein, Mohamed Kassem, Mahesh Marina, and Kimon Kontovasilis. «FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks». In: (Nov. 2016), pp. 427–441. DOI: 10.1145/2999572.2999599 (cit. on pp. 16, 32).

[11] Chin-Ya Huang, Chung-Yin Ho, Navid Nikaein, and Ray-Guang Cheng. «Design and Prototype of A Virtualized 5G Infrastructure Supporting Network Slicing». In: *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)* (2018), pp. 1–5 (cit. on pp. 18, 20, 21).

[12] Gines Garcia-Aviles, Marco Gramaglia, Pablo Serrano, Francesco Gringoli, Sergio Fuente-Pascual, and Ignacio Labrador Pavón. «Experimenting with open source tools to deploy a multi-service and multi-slice mobile network». In: *Comput. Commun.* 150 (2020), pp. 1–12 (cit. on p. 19).

[13] Muhammad Tahir Abbas, Talha Ahmed Khan, Asif Mahmood, Javier Jose Diaz Rivera, and Wang-Cheol Song. «Introducing network slice management inside M-CORD-based-5G framework». In: (2018), pp. 1–2. DOI: 10.1109/NOMS.2018.8406113 (cit. on pp. 20, 21).

[14] Prodromos-Vasileios Mekikis, Kostas Ramantas, Angelos Antonopoulos, Elli Kartsakli, Luis Sanabria-Russo, Jordi Serra, David Pubill, and Christos Verikoukis. «NFV-Enabled Experimental Platform for 5G Tactile Internet Support in Industrial Environments». In: *IEEE Transactions on Industrial Informatics* 16.3 (2020), pp. 1895–1903. DOI: 10.1109/TII.2019.2917914 (cit. on p. 22).

[15] *Smart End-to-end Massive IoT Interoperability, Connectivity and Security*. URL: https://www.semiotics-project.eu (cit. on p. 23).

[16] Florian Kaltenberger, Aloizio P Silva, Abhimanyu Gosain, Luhan Wang, and Tien-Thinh Nguyen. «OpenAirInterface: Democratizing innovation in the 5G Era». In: *Computer Networks* 176 (2020), p. 107284 (cit. on p. 28).

[17] *openairinterface5G*. URL: https://gitlab.eurecom.fr/oai/openairinterface5g/blob/master/doc/FEATURE_SET.md (cit. on pp. 28–30).

[18] *OAI 5G RAN PROJECT GROUP*. URL: https://openairinterface.org/oai-5g-ran-project/ (cit. on p. 30).

[19] Mosaic5G. *FLEXRAN: First Open-source Implementation of a Flexible and Programmable Platform for Software-Defined Radio Access Networks*. URL: https://mosaic5g.io/flexran/ (cit. on pp. 32, 34).

[20] *OpenAirInterface Core Network: Recent enhancements in OAI EPC*. URL: https://www.openairinterface.org/docs/workshop/1stOAINorthAmericaWorkshop/Training/GAUTHIER-OAI_WS_2019_NJ_en.pdf (cit. on p. 36).

[21] Benjamin Moseley and Shai Vardi. «The Efficiency-fairness Balance of Round Robin Scheduling». In: *Operations Research Letters 50* 1 (2022), pp. 20–27. (Cit. on p. 50).

[22] *tshark(1) Manual Page*. URL: `https://www.wireshark.org/docs/man-pages/tshark.html` (cit. on p. 83).