ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

Scheduling in 5G Networks: Developing a 5G Cell Capacity Simulator

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Gabriela Pereyra

en cumplimiento parcial de los requerimientos para la obtención del título de Magister en Ingeniería Eléctrica.

Directores de Tesis

Claudina Rattaro ..... Universidad de la República
Pablo Belzarena ..... Universidad de la República

Tribunal

José Acuña ..... Universidad de la República
Víctor González Barbone ..... Universidad de la República
Alberto Castro ..... Universidad de la República

Director Académico

Pablo Belzarena ..... Universidad de la República

Montevideo
Tuesday 16th November, 2021

Scheduling in 5G Networks: Developing a 5G Cell Capacity Simulator, Gabriela Pereyra.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 124 páginas.
Compilada el Tuesday 16th November, 2021.
http://iie.fing.edu.uy/

Sean los orientales tan ilustrados como valientes.

José Gervasio Artigas

This page has been intentionally left in blank

Acknowledgement

First of all, I would like to thank my two thesis directors, Claudina Rattaro and Pablo Belzarena. This work wouldn't have been possible without your support. Second, I would like to thank the SCAPA and UdelaR for giving me the opportunity of having this experience. I also want to give a special acknowledgement to Víctor Gonzalez Barbone, for sharing recommendations about Python coding documentation tools, and a few reflections about how to maintain motivation on pandemic times, when I was almost completely lost.

I want to thank my friends, who are always asking how am I doing with this and were there during the hard times.

Finally, I would like to thank my husband, for being always there, in the goods and in the bads. So many big things have happened during this years that I think I have changed, but the only thing that has not changed is that he is still on my side.

This page has been intentionally left in blank

To my husband who walked along the way with me during this entire process.

This page has been intentionally left in blank

Resumen

La quinta generación de comunicaciones móviles (5G) se está convirtiendo en una realidad gracias a la nueva tecnología 3GPP (3rd Generation Partnership Project) diseñada para cumplir con una amplia gama de requerimientos. Por un lado, debe poder soportar altas velocidades y servicios de latencia ultra-baja, y por otro lado, debe poder conectar una gran cantidad de dispositivos con requerimientos laxos de ancho de banda y retardo. Esta diversidad de requerimientos de servicio exige un alto grado de flexibilidad en el diseño de la interfaz de radio. Dado que la tecnología LTE (Long Term Evolution) se diseñó originalmente teniendo en cuenta la evolución de los servicios de banda ancha móvil, no proporciona suficiente flexibilidad para multiplexar de manera óptima los diferentes tipos de servicios previstos por 5G. Esto se debe a que no existe una única configuración de interfaz de radio capaz de adaptarse a todos los diferentes requisitos de servicio. Como consecuencia, las redes 5G se están diseñando para admitir diferentes configuraciones de interfaz de radio y mecanismos para multiplexar estos diferentes servicios con diferentes configuraciones en el mismo espectro disponible. Este concepto se conoce como Network Slicing y es una característica clave de 5G que debe ser soportada extremo a extremo en la red (acceso, transporte y núcleo). De esta manera, las Redes de Acceso (RAN) 5G agregarán el problema de asignación de recursos para diferentes servicios al problema tradicional de asignación de recursos a distintos usuarios.

En este contexto, como el estandar no describe cómo debe ser la asignación de resursos para usuarios y servicios (quedando libre a la implementación de los proveedores) se abre un amplio campo de investigación. Se han desarrollado diferentes herramientas de simulación con fines de investigación durante los últimos años. Sin embargo, como no muchas de estas son libres, fáciles de usar y particularmente ninguna de las disponibles soporta Network Slicing a nivel de Red de Acceso, este trabajo presenta un nuevo simulador como principal contribución.

Py5cheSim es un simulador simple, flexible y de código abierto basado en Python y especialmente orientado a probar diferentes algoritmos de scheduling para diferentes tipos de servicios 5G mediante una implementación simple de la funcionalidad RAN Slicing. Su arquitectura permite desarrollar e integrar nuevos algoritmos para asignaciòn de recursos de forma sencilla y directa. Además, el uso de Python proporciona suficiente versatilidad para incluso utilizar herramientas de Inteligencia Artificial para el desarrollo de nuevos algoritmos. Este trabajo presenta los principales conceptos de diseño de las redes de acceso 5G que se tomaron como base para desarrollar la herramienta de simulación. También describe decisiones de diseño e implementación, seguidas de las pruebas de validación ejecutadas

y sus principales resultados. Se presentan además algunos ejemplos de casos de uso para mostrar el potencial de la herramienta desarrollada, proporcionando un análisis primario de los algoritmos tradicionales de asignación de recursos para los nuevos tipos de servicios previstos por la tecnología. Finalmente se concluye sobre la contribución de la herramienta desarrollada, los resultados de los ejemplos incluyendo posibles líneas de investigación junto con posibles mejoras para futuras versiones.

Abstract

The fifth generation of mobile communications (5G) is already becoming a reality by the new 3GPP (3rd Generation Partnership Project) technology designed to solve a wide range of requirements. On the one hand, it must be able to support high bit rates and ultra-low latency services, and on the other hand, it should be able to connect a massive amount of devices with loose bandwidth and delay requirements. Such diversity in terms of service requirements demands a high degree of flexibility in radio interface design. As LTE (Long Term Evolution) technology was originally designed with Mobile Broadband (MBB) services evolution in mind it does not provide enough flexibility to multiplex optimally the different types of services envisioned by 5G. This is because there is not a unique radio interface configuration able to fit all the different service requirements. As a consequence, 5G networks are being designed to support different radio interface configurations and mechanisms to multiplex these different services with different configurations in the same available spectrum. This concept is known as Network Slicing, and is a 5G key feature which needs to be supported end to end in the network (Radio Access, Transport and Core Network). In this way 5G Radio Access Networks (RAN) will add the resource allocation for different services problem to the user resource allocation traditional one.

In this context, as both users and services scheduling is being left to vendor implementation by the standard, an extensive field of research is open. Different simulation tools have been developed for research purposes during the last years. However, as not so many of them are free, easy to use, and particularly none of the available ones supports Network Slicing at RAN level, this work presents a new simulator as its main contribution.

Py5cheSim is a simple, flexible and open-source simulator based on Python and specially oriented to test different scheduling algorithms for 5G different types of services through a simple implementation of RAN Slicing feature. Its architecture allows to develop and integrate new scheduling algorithms in a easy and straightforward way. Furthermore, the use of Python provides enough versatility to even use Machine Learning tools for the development of new scheduling algorithms. The present work introduces the main 5G RAN design concepts which were taken as a baseline to develop the simulation tool. It also describes its design and implementation choices followed by the executed validation tests and its main results. Additionally this work presents a few use cases examples to show the developed tool's potential providing a primary analysis of traditional scheduling

algorithms for the new types of services envisioned by the technology. Finally it concludes about the developed tool contribution, the example results along with possible research lines and future versions improvements.

Contents

This page has been intentionally left in blank

Chapter 1

Introduction

This chapter introduces the document in terms of the main goals presented for this thesis, along with its contribution. Related publications are also introduced as a means to note the presented work's context and value. Finally, this chapter ends with a brief description of the following chapters in this thesis document.

1.1 Motivation

Mobile networks have had an important evolution during the last three decades. From the possibility of making a voice call without using a wired terminal, to enabling a network of wireless sensors located in different places to report environment data to a central system, which is a reality nowadays. Technology has grown and evolved along with the use cases and human needs. Twenty years ago, people was stunned by the possibility to send an email. Today, the idea of a day without internet is unthinkable, and wireless internet is an important piece of the cake. Even when we are at home, with an ADSL connection and maybe no mobile broadband service, we would still feel the absence of WiFi to connect our smartphones. Today, even in different magnitudes, wireless networks are simply part of our lives.

In response to the increase of use of mobile broadband services, and the new use cases expected to take place in the market for the next years, a new mobile generation is being standardized by 3GPP (3^{rd} Generation Partnership Project) from Release 15 (2018): the 5^{th} Generation of Mobile Communications, or 5G for short. As wireless networks work using radio-frequency spectrum as a resource to transport data bits and the later is a scarce supply, radio access resources allocation for different users and services is a central topic, specially when using licensed spectrum. Moreover, as the standard does not define how scheduling of resources should be made, being the topic left to vendor implementation, an extensive field of research is open. As users scheduling constitutes a very rich research topic specially when it comes to cellular networks, there has been a lot work about it during the last years. However, as mentioned in [22], given the different services 5G networks are envisioned for, there is not an unique radio configuration able to meet with

so different requirements in an optimal way. In consequence, 5G networks should allow the coexistence of different "virtual networks", configured according to the services which it is used for, sharing physical resources as optimally as possible. This concept is known as Network Slicing, and is a 5G key feature which must be supported end to end in the network, including the RAN (Radio Access Network), as a way to enable different services multiplexing on the same network resources.

In this context, this thesis focuses on 5G networks radio resources allocation, particularly thinking in a multi-service capable network. The main objective of this work was to make a primary analysis of resource allocation possibilities and considerations for the different types of services envisioned by the technology evolution. As at this work's beginning there was not a simple multi-Slice 5G simulation tool for cell capacity analysis oriented to scheduling and no real network's test data available, a high level cell capacity simulator was developed as this thesis main contribution. This work is intended to provide a simulation tool to introduce 5G multi-Slice scheduling research.

1.2 Related Work

Although there is well known history about users scheduling research, given the novelty of the Network Slicing concept, research about this particular topic is quite new, as can we see in [35], [34] and [21]. In [35] different options and existing challenges for RAN Slicing implementation are identified. The authors in [34] purpose a framework for the support of RAN Slices based on the analysis of the impact in the radio interface protocols. In [21] a framework fully compliant with 3GPP vision is proposed to enforce network slicing featuring radio resources abstraction. Furthermore, given the recent progress in AI (Artificial Intelligence), integration of this kind of concepts to network research topics seems natural at this point, given the dynamic nature of mobile traffic. Authors in [43] propose a slicing method based on DRL (Deep Reinforcement Learning).

None of the referenced works proposes the use of traditional scheduling algorithms for multiplexing different services in a cell. Furthermore, to the best of this work's author knowledge, there hasn't been extended public research about the use of traditional algorithms for other traffic profiles different than MBB (Mobile Broadband).

Although at the moment of start this work there was several free 5G ready simulation tools available, none of them was chosen for this thesis purpose given their limitations in terms of easiness of use, 5G features support and possibility of new schedulers implementation.

At this moment there are different types of simulators available for 5G. In the one hand, Network Simulators have great level of detail and a wide range of configurable options giving a good reference to compare at the time of validation. A good example of this is the *5G-LENA* simulator [32], (which by the way was the chosen tool for validating the developed one). Its last version was released in March/2021 and was built as a module (*nr*) for the *ns3* simulator. It is strongly based in its predecessors, the *Lena* module [9], and the *mmWaves* module [23],

both also in *ns3*. However these modules present some disadvantages: the high degree of complexity and processing capacity needed to configure and run a simulation. The first is not a problem if one is a C++ developer and has experience with ns-3, but the second is unavoidable because of the simulator's nature. These modules often implement layer by layer most of the procedures described by the 3GPP standard, so a simple 10 minutes simulation with high bandwidth and several users can take hours in a standard PC. Additionally, although 5G-LENA supports many NR (New Radio) features, the current version of this module (NRv1.1) does not implement mini-slots and network slicing scheduling.

On the other hand, System Level Simulators could be a good option, however it often incurs in a high degree of simplification to cover a wide range of cells with an affordable resource use. Some examples are the *Vienna Simulator* [24, 33] and *Simu5G* [25, 26]. The first is a Matlab tool, again, based on its LTE (Long Term Evolution) predecessor (the *Vienna LTE simulator*). The latest version of Viena (2020) does not support key NR features like mini-slot scheduling, network slicing, mmWave propagation models, and 256-QAM (256 Quadrature Amplitude Modulation). Besides, the possibility to perform uplink simulations and non-full buffer traffic models is not included. *Simu5G* is based on OMNeT++ framework and written in C++. It simulates the data plane of the 5G RAN and core network. It supports FDD (Frequency Division Duplex) and TDD (Time Division Duplex) modes, Dual Connectivity, Carrier Aggregation, and different numerologies. However, according to its last version (1.1.0), it does not support network slicing or mini-slot.

Another available simulation tools are *5G-K-Sim* [5,20] and *SyntheticNET* [44]. The first one is a complete C++ tool that includes link-level, system-level, and network-level simulations. However it does not support end-to-end network slicing. *5G-K-Sim* was developed at the earliest stages of the 5G standardization process and is non-fully standard compliant. On the other hand, *SyntheticNET* [44] is a Python simulator focused on modeling a realistic handover process. It supports different numerologies and mmWaves, but is not clear if it supports other 5G features.

Finally, it is important to mention *OpenAirInterface* [17, 18]. It is an open source software running on general-purpose processors which supports many of NR specifications. Although it has limitations when emulating large networks, it could be a good tool to validate new proposals in real testbeds.

Given the thesis objective, available tools were not enough to provide a meaningful analysis. On the one hand, most system level simulators were not supporting essential features like RAN Slicing, or 256QAM modulation. Furthermore, it hasn't too much different possibilities in terms of scheduling algorithms. On the other hand, network simulators was too much in terms of complexity for the pretended results, even not supporting features like RAN Slicing. With this ideas in mind a new simulation tool was developed. The general design goal was to keep it as simple and flexible as possible mostly when it comes to scheduler implementation.

1.3 Thesis Contribution

As mentioned before, this thesis main contribution is a free simulation tool for 5G high level cell capacity analysis focused on resource allocation between different users and services with very different requirements. The simulation tool was called *Python 5G Scheduler Simulator* or *Py5cheSim* for short, in honor to Python as the language used, and the fact that it was designed to support the development of new 5G scheduling algorithms to study. It provides a framework for new 5G scheduling algorithms development and evaluation and by the moment is the only available simulation tool supporting RAN Slicing. Other 5G simulation tools available are mentioned and referenced, highlighting the reasons why they were not considered for this work.

This thesis also presents a brief study about 5G radio access technology main design aspects and possibilities, along with a few examples of traditional schedulers primary performance analysis for users and services resource allocation. Finally, this work concludes about the developed tool and possible design considerations on 5G schedulers implementation considering resource allocation between different users and services which the technology is designed for. This work is intended to be useful as a starting point for future and more detailed research about scheduling in 5G networks and beyond.

Finally, this work has produced the next publications:

- *Py5cheSim: a 5G Multi-Slice Cell Capacity Simulator.* XLVII Confererencia Latinoamericana en Informatica CLEI 2021 [2].
- *A 5G multi-Slice cell capacity framework.* SIGCOMM N2Women Workshop 2021 [1].

Current *Py5cheSim* version along with its code documentation can be found in [10] [1].

1.4 Document Organization

The presented document is organized as following:

First, main 5G service requirements and resulting radio access technology design aspects are introduced, focusing on the key concepts related to radio resource allocation. Then, a concise description of the developed tool for cell capacity analysis is presented. *Py5cheSim* features and its main design considerations along with the standard procedures abstraction are shown to understand the application field of the developed tool.

Chapter 4 shows the *Py5cheSim* validation results using different available simulation and performance estimation tools. Different traditional schedulers results are shown here only for scheduler validation purposes. Chapter 5 presents different

[1] https://iie.fing.edu.uy/investigacion/grupos/artes/proyectos/inteligencia-artificial-aplicada-a-redes-5g/

use cases for the developed simulation tool highlighting the information provided and analysis results.

Finally, chapter 6 concludes the work in terms of the developed tool contribution, application scenarios and possible evolution. It also presents the main results obtained by testing traditional scheduling algorithms on 5G networks, and concludes about future schedulers design considerations.

This page has been intentionally left in blank

Chapter 2

From 4G to 5G Networks

This chapter presents a brief introduction of the 5G's most transcendent concepts for this thesis. Although there is a lot to learn and to say about 5G networks, this introduction will be more oriented to describe some of the basic ideas behind the technology design which have most to do with the thesis objective. The best way to understand 5G networks is to start from 4G networks, because most of the 5G network design starts from an evolution of 4G networks to meet the 5G new requirements.

The chapter is organized as follows: First, 4G networks main design aspects are introduced as a baseline to understand 5G's. Then, it describes service requirements evolution which justify 5G networks. After that, it shows an introduction to 5G main design aspects and technological enablers.

2.1 4G Networks, a brief Introduction

The 4^{th} generation of mobile communications, or 4G for short, was standardized by 3GPP from Release 10 during the last decade, and most countries in the world have deployed it during the last years. LTE (Long Term Evolution) technology was designed to meet the RAN (Radio Access Network) constraints, which along with the EPC (Evolved Packet Core) supports the ITU requirements for IMT-Advanced. Those requirements were mostly oriented to MBB (Mobile Broadband) communications attending to the expected traffic needs at that time, as high bitrate, low delays, and QoS (Quality of Service) support, for example.

A simple, almost flat architecture was envisioned for the network, simplifying in a good way the Core features and protocols from the previous generation's design. At RAN level, the simplification was even higher by using the multiple access technologies OFDMA (Orthogonal frequency-division multiple access) for DL (Downlink) and SC-FDMA (Single Carrier Frequency for UL (Uplink). As 5G RAN also uses OFDMA, at least for the first phase of the standard, the LTE's layer 1 and 2 design will be introduced from here on.

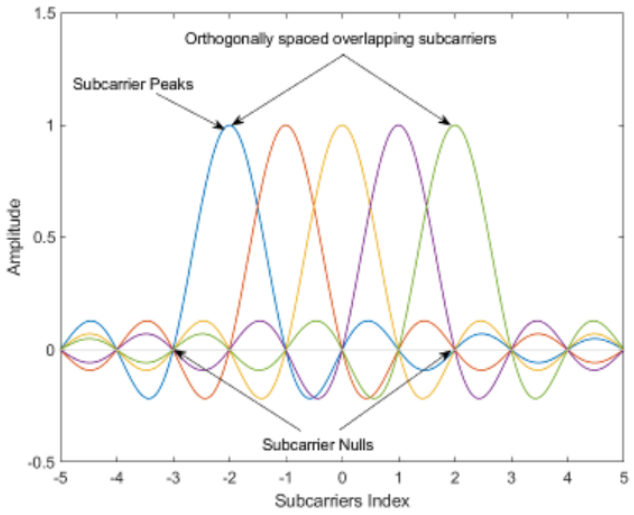

Figure 2.1: OFDMA orthogonal sub carriers in frequency domain example from [6].

2.1.1 LTE RAN Basic Design Aspects

As it's been said before, LTE uses OFDMA multiple access technology in DL. In OFDMA the available spectrum is divided in orthogonal subcarriers equally spaced. Orthogonality here means that even if the subcarrier spacing is minimal there is no interference between them, as can be seen in figure 2.1.

The bits flows to transmit are modulated with an appropriate MCS (Modulation and Coding Scheme) using one sub carrier for each symbol. Then IFFT (Inverse fast Fourier transform) is applied to the symbols in parallel, cyclic prefix is added to reduce ISI (Inter-symbol interference), D/A (Digital to Analogue) conversion is applied, the signal is moved to the allocated band and transmitted. In reception, the procedure is basically the opposite. The received signal is moved to base band, sampled and transformed to the frequency domain by applying the FFT (Fast Fourier Transform), and by detecting phase and amplitude of each symbol the original bits sequence can be obtained. The later can be summarized in figure 2.2.

Orthogonality is assured by making the SCS (Subcarrier spacing) equal to the symbol duration inverse, which is shown in figure 2.1. There, the SCS can be seen as the difference between two consecutive subcarrier peacks, and it must be equal to the symbol duration inverse to make the peack value for a sub carrier to be located in the null values for all the others in the FFT. To select the SCS, the *Doppler effect* caused by UE moving must be considered. SCS must be high enough for the signal shift to not be meaningful compared to it. In LTE 15 kHz of SCS was chosen as is high enough to make the frequency shift caused by an UE moving at 350 km/h not meaningful. On the other hand, the *ISI* possible effects must be considered when choosing the SCS. ISI is produced by multipath fading, which is unavoidable in wireless networks. Several copies of the same signal will arrive at RX at different times producing a shift on the symbols in time domain.

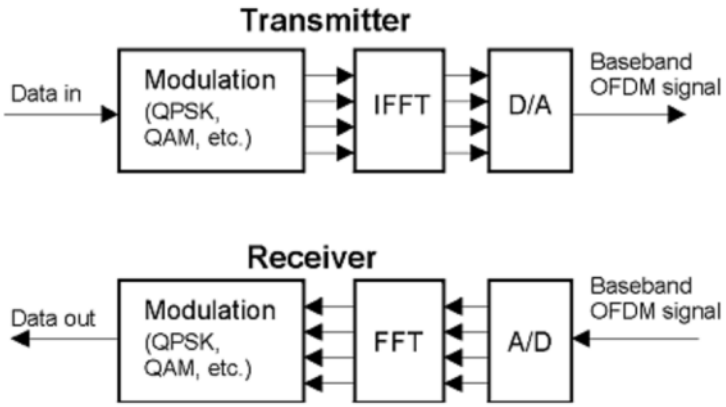

Figure 2.2: OFDMA TX and RX scheme [7].

The symbol duration must be high enough for this shift to be not meaningful. This adds a new constraint to define SCS, so there is a compromise between the two things, to be robust enough to Doppler effect and ISI. As for the later, LTE also adds the use of Cyclic Prefix (CP), which basically consists in adding the last symbol samples at the beginning of it, so it can be recovered even if it was slightly shifted in time.

2.1.2 Radio Interface Architecture

Figure 2.3 shows LTE radio interface protocol stack and related 3GPP TS (Technical Specifications). Information flows between layers happen by the use of channels and signals. There are different types of channels depending on the type of information to transmit, and how is going to be processed. Putting in a nutshell, LTE handles:

- Bearers: UE service and signalling flows, which are mapped in Logical Channels.
- Logical Channels which are mapped to Transport Channels at the MAC layer according to the way they will be transmitted. Users data are mapped to DL-SCH and UL-SCH for DL and UL respectively.
- Transport Channels which are mapped to Physical Channels at the first sub layer of the Physical layer.
- Physical Channels which are mapped to the physical resources in the frequency-time resource grid.

2.1.3 Resource Allocation

The physical resources are basically the subcarriers mentioned before in the used band during time. *In time* domain, for each subcarrier 7 (or 6 in case of using

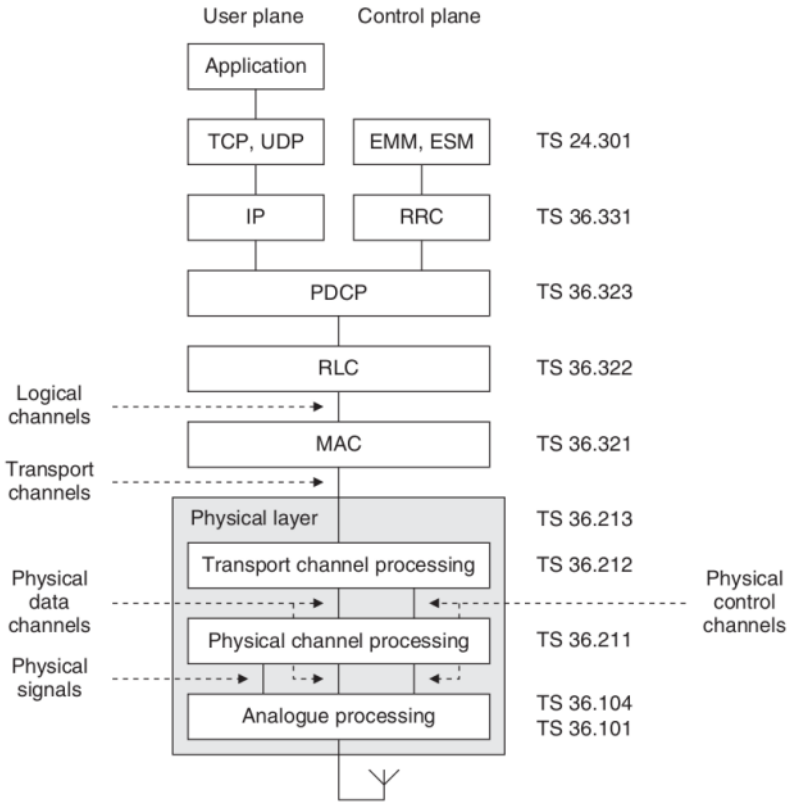

Figure 2.3: LTE radio interface protocol stack and associated 3GPP TS (Technical Specifications) [7].

extended cyclic prefix) symbols are transmitted in a slot, and 2 slots in a sub frame of 1 ms, which is part of a 10 ms frame, as can be seen in figure 2.4.

In frequency domain the subcarriers are grouped in PRB (Physical Resource Block), which consists in 12 consecutive subcarriers. In this way, physical resources are organized in a frequency-time grid, and allocation for each UE is done in a sub frame granularity, which in this case is 1 ms. This time interval is also called TTI (Transmission Time Interval). That is, for each TTI, the smallest amount of resources that can be allocated for a UE is 1 PRB. So for each TTI the cell can roughly allocate the next amount of symbols to a UE for DL:

$$AllocatedPRBs \times 12 \times (14 - PDCCHsymbols - PHYsignalsSymbols). \quad (2.1)$$

PDCCH is the Physical Downlink Control Channel, and is basically used to transmit Downlink Control Information (DCI), as resource allocation information, UL ACK (Acknowledgement), etc. Typically 3 symbols are used for this channel, but it can be configured differently. Also some symbols are defined in DL to transmit PHY (Physical Layer) signals to the UEs for synchronization, cell detection and channel estimation procedures.

The resource allocation to each UE at each TTI along with the MCS election

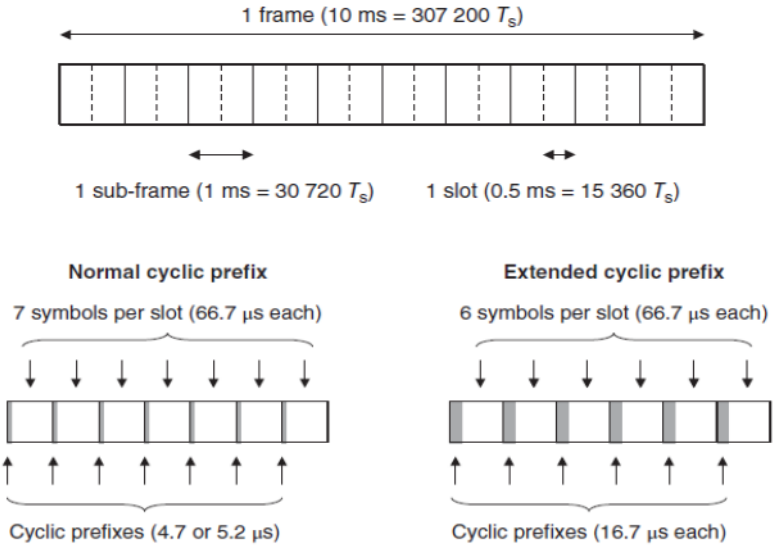

Figure 2.4: LTE frame and slot structure using normal (left) and extended (right) cyclic prefix [7].

is done by the cell's *scheduler*. It takes into account different information of each UE as Channel state, transmission buffer size, QoS requirements, possibility to use different MIMO schemes, acknowledgments, etc. The standard does not define how the scheduler algorithm works, so the later is vendor's implementation dependent. Several algorithms were studied to this propose, considering different approaches. In one side the more resources the cell gives to a UE with a good channel, the more optimally the cell resources are used, but users with a poor channel are not going to have high bitrates. On the other side, if the same amount of resources is used for all UEs, there could be UEs with few bytes to transmit taking the same resources than others with more, so the resource are not being allocated optimally. There is a compromise between optimal resource use and fairness. One of the most commonly used algorithm considering the latter is the Proportional Fair Scheduler. In this case, allocation is done as a function of the relation between the possible throughput a UE can obtain, and the past throughput it had, so if there have not been allocated resources for this UE in the past, there is more chance to allocate in the present.

More detailed information about LTE can be found in [7].

2.2 4.5G Networks and the Requirements Evolution

Although the LTE standard begins from the 3GPP Release 8, in terms of compliance with the IMT-Advanced requirements the 4th Generation Mobile Communications starts with Release 10. However, as the technology changed from Release 8, in a commercial sense 4G started from 3GPP Release 8. That is why the following capacity and features enhancements were commercially classified as 4.5G or LTE-

Advanced. This enhancements covers all the necessary technology improvements to enhance capacity mainly in terms of user and cell throughput, but also in terms of resource use when M2M (Machine to Machine) devices are connected. For the former, features like 256QAM (Quadrature Amplitude Modulation) use, MIMO (Multiple Input Multiple Output), and Carrier Aggregation were introduced. For the later, two LTE optimizations were introduced in Release 13: LTE-M (LTE for Cat M devices) and NB-IoT (Narrow Band Internet of Things). In the following, this enhancements will be briefly introduced, as it will be also part of the 5G standard.

2.2.1 Carrier Aggregation

By using this feature, a UE can use several component carriers (CC) to communicate with the network instead of one. This increases throughput capacity because the UE can use resources from more than one carrier as can be seen in figure 2.5. While blue and red UE uses only RBs from blue and red carrier, the white UE can use RB from the three carriers.

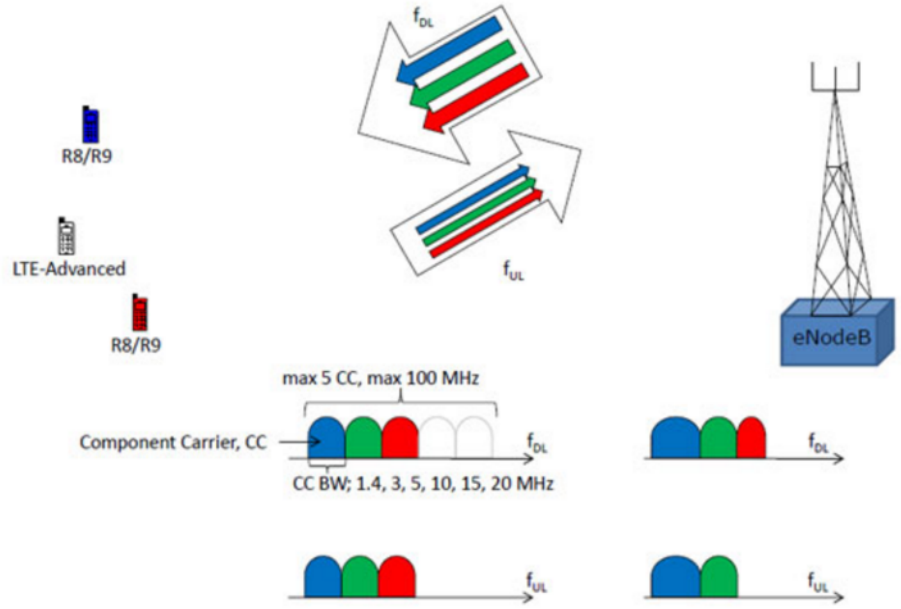

Figure 2.5: Carrier aggregation resource use example [16].

The carriers can be in the same band, in a contiguous location or not, or in a different band. The amount of CC a UE is allowed to use depends on the UE and network capabilities and has evolved with the 3GPP Releases. In this way the UE connects to different cells, one of them will be called primary component carrier (PCC), which will handle the RRC (Radio Resource Control) connection, and the rest will be called secondary component carriers (SCC), and will handle just UP data, as can be seen in figure 2.6. As black UE supports Carrier Aggregation in

blue, red and green carrier, as is located in the respective cell's coverage area, it uses resources from the three carriers. However, as the RRC connection is handled only by the blue cell, this will be the it's PCC, and the other cells only handle user data for this UE.

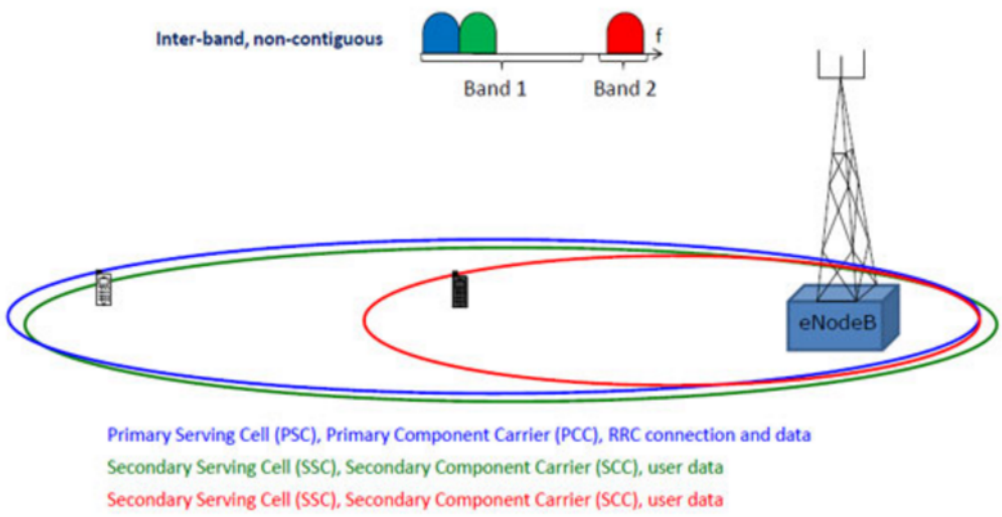

Figure 2.6: Carrier aggregation PCC and SCC example [16].

2.2.2 256QAM

By using this feature a new MCS table is introduced, considering the use of 256QAM modulation. As with the later releases the maximum modulation scheme supported was 64QAM, the introduction of 256QAM implies an important throughput capacity increase (in good radio conditions), because there are more modulated bits/symbol as can be seen in figure 2.7. While 64QAM allows to carry six bits by symbol, 256QAM allows eight.

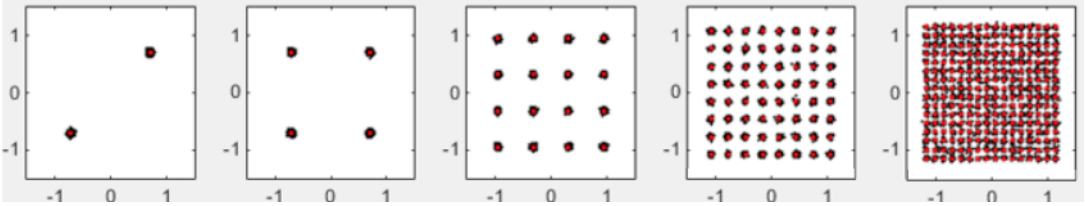

Figure 2.7: Different modulation schemes constellation diagram examples from [12]. From left to right the figure shows BPSK, QAM, 16QAM, 64QAM, and 256 QAM constellation examples.

2.2.3 MIMO

Although MIMO use was introduced in Release 8, from Release 10 several enhancements were added to this feature, paving the way for massive MIMO, a key enabler for 5G. There are several possible uses for multiple antennas systems:

- **Diversity**: Transmitting/Receiving the same symbols by different antennas with a phase difference so that signals interfere positively between each others to improve communication robustness.
- **Spatial multiplexing or MIMO**: Transmitting/Receiving different symbols by different antennas at the same frequency-time to improve bitrate. Here, there are two possibilities: SU-MIMO (Single User MIMO), when the different symbols are transmitted for the same UE, and MU-MIMO (Multiple User MIMO), when the later are transmitted for different UEs. A system with N transmission and reception antennas using this techniques is called NxN MIMO, and N is also the amount of layers, i.e. flows the system can handle using the same time-frequency resources. Note that it is possible only if the channel characteristics allow it. In mathematical terms, this occurs if the rank of the matrix which represents the channel between N transmitting and N receiving antennas is N. If the rank is less than N, only a number of flows equal to this rank could be handled using the same resources.
- **Beamforming**: Transmitting/Receiving the same symbols by different antennas with a combination of phase and amplitude that modifies the resulting radiation pattern in a "beam form" between transmitter en receptor. This increase the communication range in terms of distance, and ends up enhancing cell coverage.

In Release 8 up to 4x4 SU-MIMO is supported for DL, and MU-MIMO only in UL. In Release 10 up to 8x8 SU-MIMO is supported for DL and up to 4x4 in UL. The next 3GPP releases introduced also several enhancements in terms of channel estimation, a key concept for this feature to properly work.

2.2.4 LTE-M and NB-IoT

During the last years the use of IoT devices is been increasing, so 3GPP in response to this, presented two LTE optimizations to support new IoT use cases: LTE-M, and NB-IoT. In this case the design goals were totally different than in LTE. IoT devices are generally simple, low cost devices which transmit small packets with low rate, but can be a lot of these connected to a cell. As LTE was primarily designed for MBB, several optimization were made to handle IoT traffic in a more optimal way.

- **LTE-M** was introduced in 3GPP Release 13 as an LTE optimization for Cat-M devices. These devices are supposed to be simpler and cheaper than a regular one, and are intended to use smaller bit rates, and support a limited set of LTE features. At a practical level, LTE-M works as a feature in the

RAN, which allows to connect Cat-M devices [1] using 6 PRB (a narrow-band) in the cell's band. The LTE-M physical channels are mapped differently using this limited resources, so the LTE cell capacity is minimally affected.

- **NB-IoT** was introduced in 3GPP Release 13, also as another LTE optimization for Cat-NB devices [2]. This devices are supposed to be even simpler and chipper than Cat-M ones, and intended to support more limited bitrates and legacy features. For example, RRC-Connected mobility is supported in LTE-M but not in NB-IoT. At a practical level, depending on the operation mode chosen NB-IoT can be enabled in a new "virtual" cell which uses only 1 PRB of the LTE legacy cell which is anchored to.

Both technologies also support specific features to improve cell coverage (Coverage Enhancement), and reduce power consumption in the device (eDRX, PSM). More detailed information about LTE-Advanced features can be found in [7], and about LTE-M and NB-IoT in [36].

2.3 5G Networks, a brief Introduction

This section briefly introduces the 5G most important concepts to this thesis. This work started during the beginning of the technology standardization process, so different documents were considered, from 3GPP Technical Recommendations to the available specifications and derived documents. In this sense this work accompanied the standardization process of the technology.

2.3.1 Mobile Requirements Evolution

As is been said before, mobile services requirements have been changed over the years together with the cellular technologies. First was voice, then sms appeared, latter mobile broadband (MBB). During the past 2 decades the required bitrates for MBB have presented a sustained growth, mainly due to high definition video services, together with the coding and electronic component capacity needed. LTE-Advanced was designed to support high bitrate MBB services, but not to support another kind of services like massive IoT devices connections or self driving cars for example. However, during the last years those kind of services are becoming more a reality than a fiction. Nowadays there are IoT devices using legacy cellular technologies (GPRS, WCDMA, LTE), other technologies (like LoRA, IEEE 802.15, etc.), and LTE optimizations mentioned before to support it. However, neither LTE-M's nor NB-IoT's physical layer is prepared for a real massive amount of connected devices. Also, new services that require ultra low delay, as self driving cars, or AR (Agumented Reality) are still far from been a reality using LTE-Advanced because of the minimum transmission time it handles (TTI = 1 ms). Some applications will require it lower.

[1] IoT devices supporting 3GPP LTE-M technology.
[2] IoT devices supporting 3GPP NB-IoT technology.

Attending to the requirement evolution mentioned before, a new generation of mobile communication is been standardized from Release 15. The 5th Generation Mobile Communication, or 5G, is been designed to tackle the new requirements gathered in IMT-2020 ITU recommendations [4]. In the figure 2.8 IMT-2020 services classification is shown. There are basically three types of services considered:

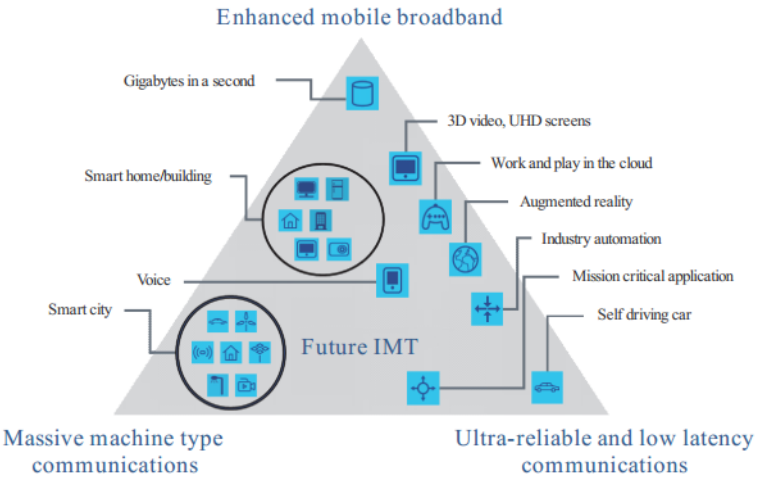

Figure 2.8: IMT-2020 different services classification from [4].

- **eMBB** (Enhanced Mobile Bradband) which is an enhancement to the traditional MBB services with higher bitrates and UE density.
- **mMTC** (Massive Machine Type Communications) which considers now the high density of IoT devices connected to the network produced by Smart city applications for example.
- **URLLC** (Ultra Reliable and Low Latency Communications) which covers all those services with ultra low latency and high reliability needs like self driving cars and mission critical applications.

As can be also seen in 2.8, there will be services between these big uses cases like Augmented Reality, Industry Automation and voice for example. These services has intermediate requirement values. In figure 2.9 IMT-2020 requirement values are shown and compared with IMT-Advanced ones, and in figure 2.10 different type of service requirements are shown.

2.3.2 Design Key Concepts

With the later service requirements in mind, different design aspects of the last RAN and Core technologies were analyzed. Several studies have shown that there is not a unique network solution to fit all services requirements [22]. The new design should contemplate the later, presenting enough *flexibility* to adapt to the different requirement constraints. This flexibility is needed across the entire network, in the

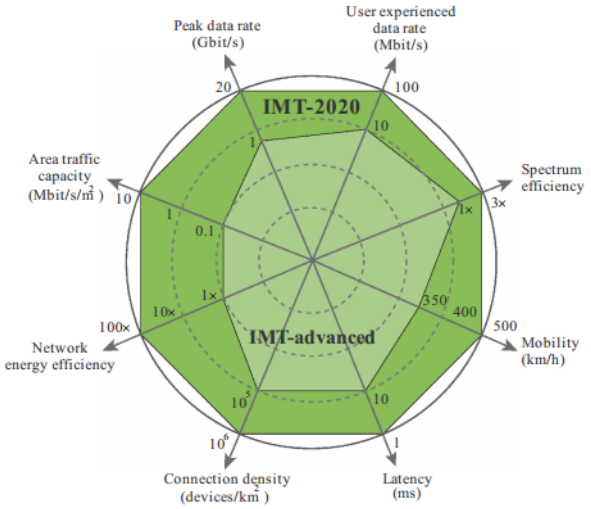

Figure 2.9: IMT-2020 vs IMT-Advanced requirements [14].

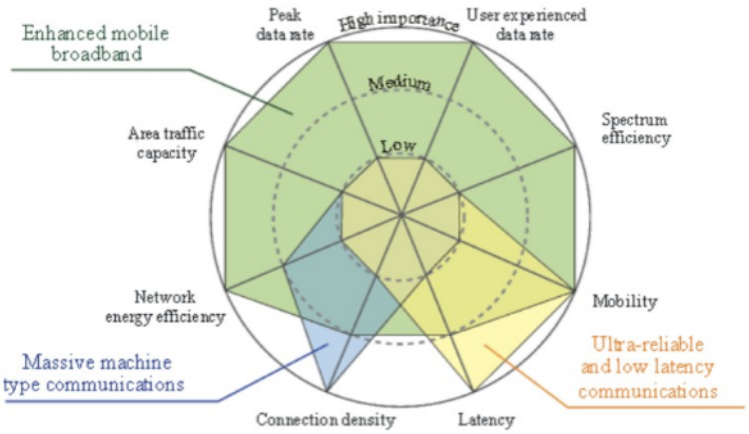

Figure 2.10: IMT-2020 different types of services requirements [14].

RAN, as much as in the Core, and also in the transport network. The services requirements are so different that one configuration optimal for one service could be a resource waste for others. As a consequence the new RAN as much as the new Core should support different configurations in different set of resources allocated for the different services. An enabler for that is the *Network Slicing* feature, which will be better described later.

In terms of RAN Physical layer design *SCS* in 5G were proposed to be variable as can be seen in [22], to adapt to different requirements. High SCS, or high numerology, as it is been called, will be configured for services with low delay requirements or high velocity for example. *MmWaves* use should be supported to provide the high bitrates expected. In this context mmWaves make reference to the spectrum frequencies above 6 GHz, although technically the millimeter waves

frequencies are higher. Different *waveforms* were also analyzed taking into account the possibility of multiplexing different SCS. The new waveform should have good confinement properties in frequency. Also, should provide enough robustness to operate with mmWaves, be compatible with MIMO, and provide low PAPR (Peak to Average Power Ratio) for UL. Another design key concept is the compatibility with legacy systems, i.e. 4G. 5G RAN should be highly compatible with LTE because there has been significant investment in LTE deployment during the last years, and the technology switch is not always fast. In this sense, the base SCS should be the same as in LTE, that is 15kHz, and LTE-Advanced features like Carrier Aggregation an MIMO should be supported. Multiple access technologies and Random Access procedures was also analyzed, mainly to enable the high amount of connections associated to mMTC services, and to allow more agile data transmission, trying to reduce the LTE overheads.

2.3.3 The Standardization Process

5G networks are being standardized by 3GPP from Release 15. The standardization process was divided in two phases. The first one, in Release 15, and the second, from Release 16. as can be seen in figure 2.11.

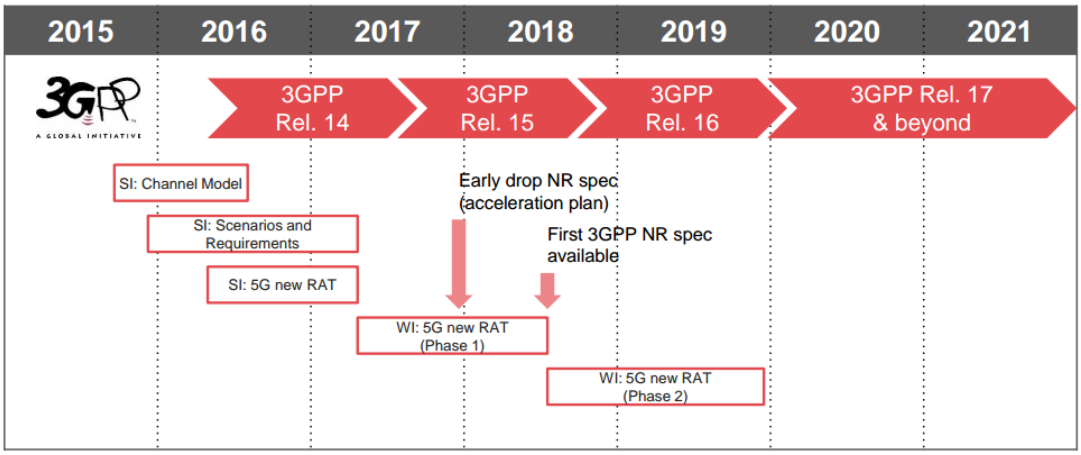

Figure 2.11: 3GPP 5G standardization process [41].

The first phase [37] introduced the new RAN and Core for 5G: NR (New Radio) and 5GC/NGC (5G Core/New Generation Core) respectively, which covered the basic design and features for eMBB services to work, and a light introduction for URLLC services. Massive MTC services were not covered in Release 15. It was released in two stages, the first one in Q2 2018 ready to support the simplest deployment option for 5G: NSA (non Stand Alone). The second stage was released in Q4 2018 supporting also SA (Stand Alone) deployments. SA deployments require the presence of the 5GC. The second phase [38] completes the specification set to support the 5G requirements specified by IMT-2020, and was submitted last year. 3GPP Release 16 was released in July 2020, extending R15 features, and

adding new ones oriented to support new use cases. Improvements was made in features like MIMO, DSS (Dynamic Spectrum Sharing) DC (Dual Connectivity), CA (Carrier Aggregation) and UE power saving. New features and deployment scenarios were also presented to support new verticals. A few examples are IAB (Integrated Access and Backhauling), NR in unlicensed spectrum, IIoT (Industrial IoT) and URLLC communication, ITS (Intelligent Transport Systems) and V2X (Vehicle to Anything) as can be seen in [15].

2.3.4 Network Architecture and Development Options

In [30] a high level description of 5G architecture, functional separation and basic features can be found. The basic network architecture is shown in figure 2.12. The AMF (Authentication and Mobility Function) is the Core entity responsible

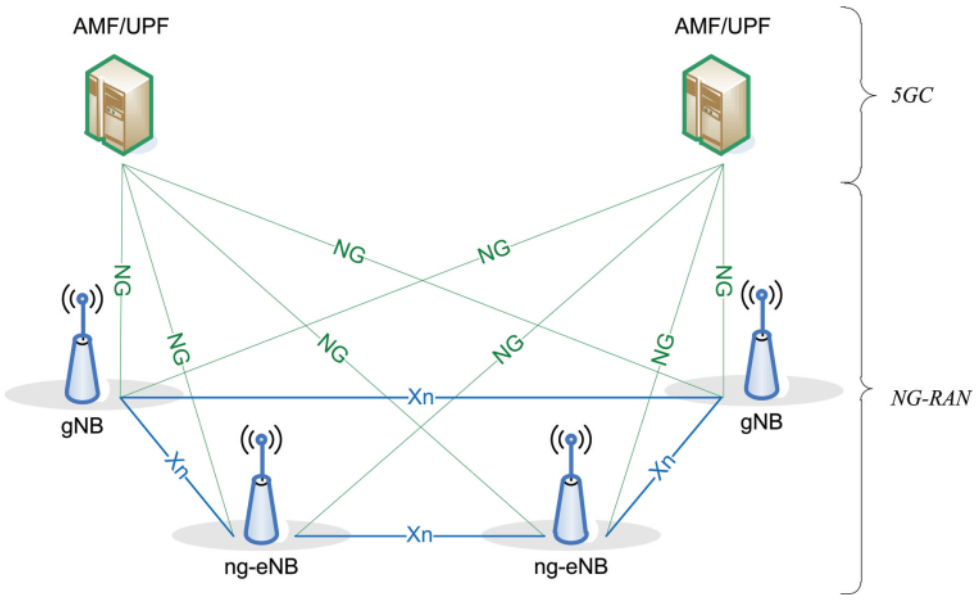

Figure 2.12: 5G basic network architecture [30].

for Authentication and Mobility support in 5G. UPF (User Plane Function) is the Core entity responsible for UP packet forwarding.

As it is been said before, 5G networks are being designed to be highly compatible with legacy 4G. As a consequence, different deployment options were considered in [39] as can be seen in figure 2.13, in terms of integration level with LTE legacy network.

2.3.5 NR: The New RAN Technology

In this subsection a brief introduction to the 5G's main aspects to this thesis is made. The main focus here will be the physical layer and how it interacts with MAC layer as for scheduling purposes.

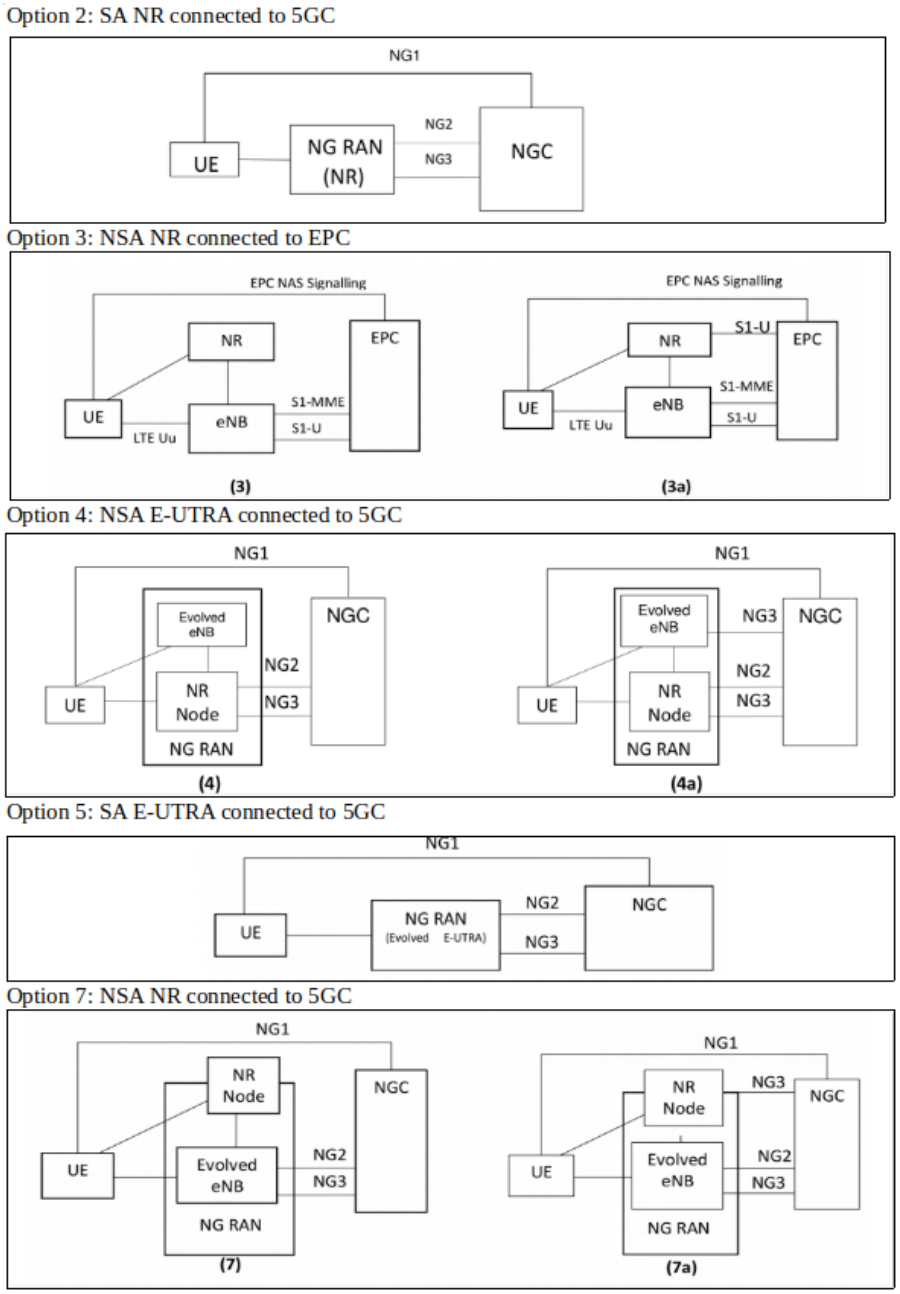

Figure 2.13: 5G different deployment options depending on 5GC availability and integration level with LTE [39].

Protocol Architecture

Figure 2.14 shows 5G RAN protocol architecture for User and Control Plane. There is almost no difference with LTE protocol stack except 5G has a specific layer for QoS support (SDAP), as mentioned in [30]. Scheduling is handled at MAC level, as in LTE, but taking into account the differences between technologies,

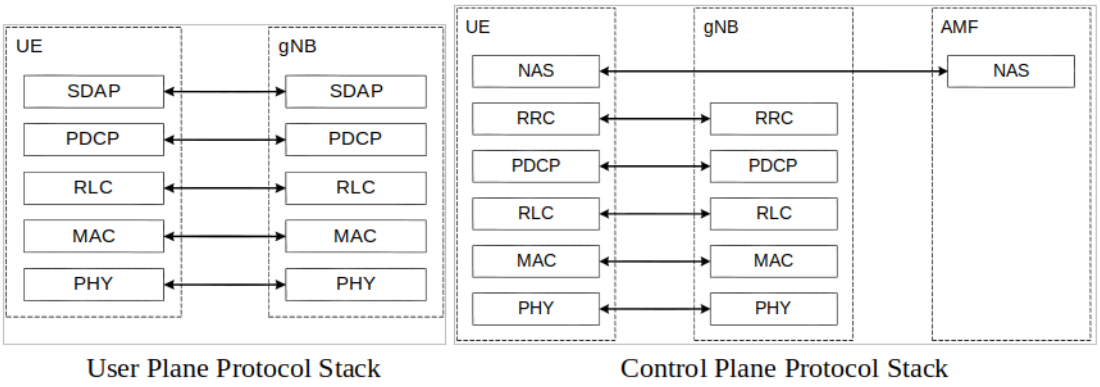

Figure 2.14: 5G RAN protocol architecture for User Plane (left) and Control Plane (right) [30].

mainly at PHY level. As in LTE, scheduling implementation is vendor dependent.

Resource Grid

At Release 15, OFDMA multiple access technology is supported to maintain high compatibility with 4G. So the resource grid is based on LTE's, but with multiple numerology support. Different SCS are supported with LTE's 15 kHz as a baseline. Other SCS can be obtained multiplying by 2^{μ} ($\mu = 0, 1, 2, ..$). In this way, different numerologies can be multiplexed on the same band as can be seen in figure 2.15.

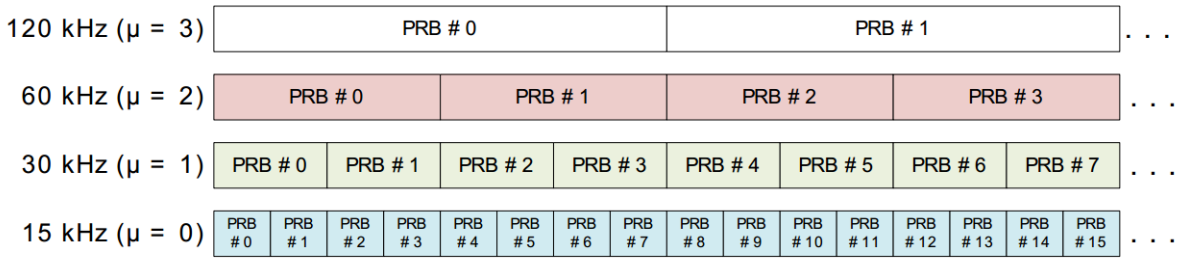

Figure 2.15: 5G Multiple numerology multiplexing [41].

As in LTE, 10 ms frame is supported at time level, and 12 subcarrier PRB at frequency level. Also, 1 ms subframe is handled. The difference with LTE is that in NR 1 slot contains always 14 symbols (with normal CP, 12 with extended CP), whatever SCS is used. Depending on the numerology, different amount of slots by subframe are handled, as can be seen in figure 2.16. Note that as SCS is bigger, symbol duration is smaller, so more slots can be contained in a subframe. The slot duration is also the *TTI*, so in this way NR handles flexible TTI to support different delay requirements. Slot level and mini-slot level scheduling is supported, being the last one specially considered for URLLC services. A mini slot is a set of 2, 4 or 7 symbols.

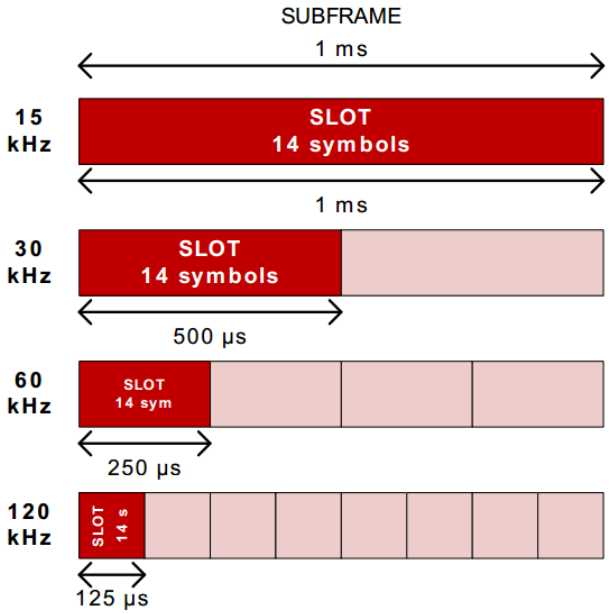

Figure 2.16: Slots by sub-frame for different numerologies [41].

As in LTE, *TDD* is also supported, but in NR TDD frame has more flexibility in terms of DL/UL resources granularity. That is, in NR one slot can be DL, UL or mixed (some symbols DL, some symbols UL) with a time gap in between, as can be seen in figure 2.17. Slot format can be static, semi-static or dynamic.
Note that in mmWaves only TDD is supported.

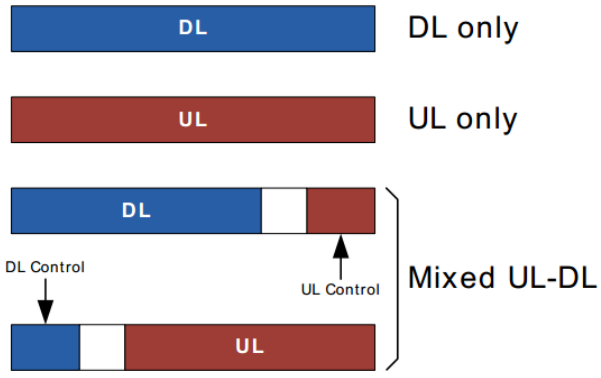

Figure 2.17: TDD Slot configuration possibilities [41].

Another new concept in NR is the BWP (Bandwidth Part). A BWP is a subset of contiguous PRB within the operation band with an specific numerology. Although more than one BWP configuration can be signalled to the UE only one in DL and one in UL is active for a user in a given instant. This concept is important for example for UEs with reduced bandwidth capability, and to support

multiplexing different numerologies in the same frequency band.

PHY Layer

Although there have not been big changes in channel definition and mapping comparing with LTE, in NR several changes in physical layer control channel and signals are introduced. With the use of beamforming for mmWaves, new physical layer signals and procedures are defined. In this way, Physical layer overhead changes compared to LTE, so 3 symbol estimation used for LTE DL is not longer valid. Explicit changes in Physical layer procedures and PDCCH are resumed in [41] and [19].

In DL, as in LTE, NR PHY Layer handles PDSCH (Physical Downlink Shared Channel) for data and PDCCH (Physical Downlink Control Channel) for DCI (Downlink Control Information) transmission. Also PBCH (Physical Broadcast Channel) is used for cell acquisition procedures. Also as in LTE PSS (Primary Synchronization Signal) and SSS (Secondary Synchronization Signal) are used for cell acquisition procedures. However, the transmission way of this signals differs from LTE, and new reference signals are added to improve channel state measurements: CSI-RS (Channel State Information Reference Signal) and TRS (Tracking Reference Signal).

Synchronization Signals are transmitted in blocks (SS Block) composed of 1 symbol PSS, 1 symbol SSS and 2 symbols PBCH. The blocks are transmitted in sequences of SS Bursts (SSB) during a 5 ms time window, with a periodicity of 20 ms. The number of possible locations in the time-frequency resource grid (L) depends on the band used. The amount of frequency and time resources dedicated to a SS block is as can be seen in figure 2.18 and 2.19.

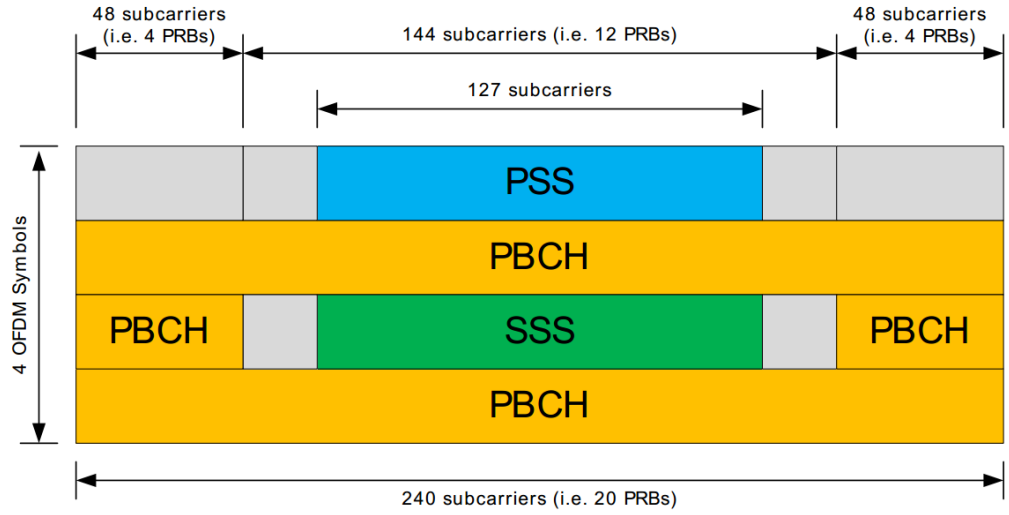

Figure 2.18: SS Block frequency resources [41].

In this way the synchronization overhead can be estimated as the rationale between number of symbols dedicated to SSB transmission during 20 ms over the

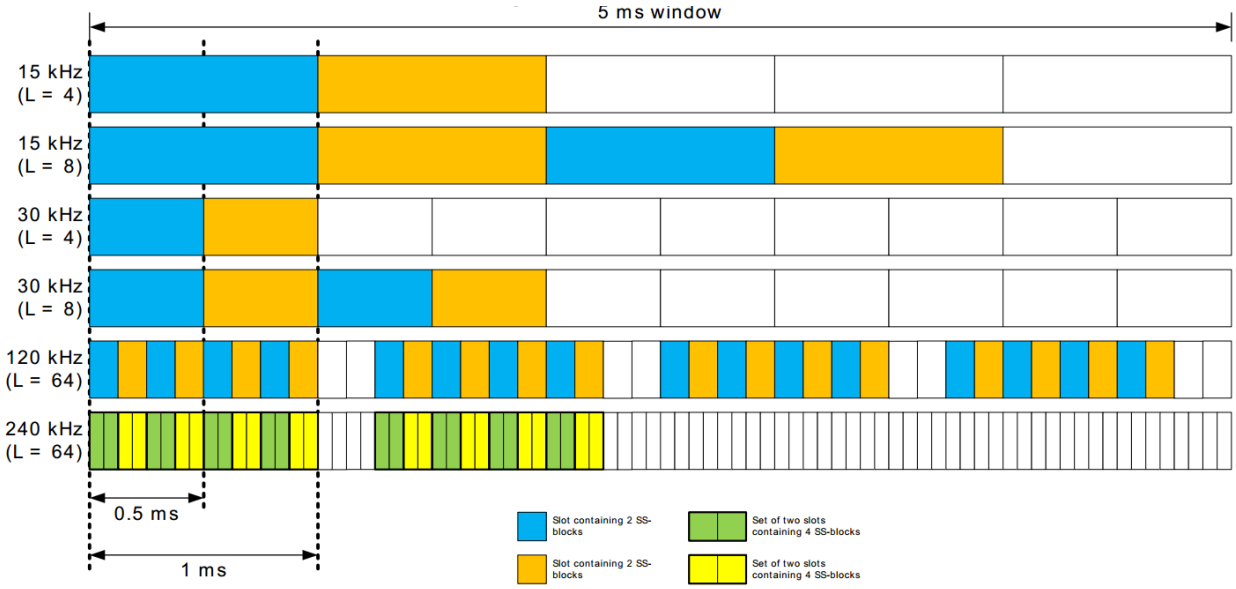

Figure 2.19: SS Block Set time resources [41].

total amount of symbols contained in the two frames. The result depends on the band and numerology used, but in the worst case it doesn't get bigger than 5% of symbols.

CSI-RS can be transmitted in different ways depending on configuration. In the worst case scenario, it takes 4 symbols/PRB with a periodicity of 4 slots, so it can imply an overhead of 0.6%. The same occurs with TRS.

As in LTE, PDCCH is used for DCI transmission, including scheduling information for DL and UL. In NR PDCCH is transmitted using a structure called CORESET (Control Resource Set). A CORESET consist of a group of REG (Resource Element Group) under a given numerology. A REG is one PRB during one symbol. In time domain it can take from 1 to 3 symbols. This along with the frequency domain resources, and starting symbol is configured by UE's specific higher layer signalling. The number of REG on which is mapped the DCI in a CORESET depends on the DCI's aggregation level handled.

In UL, as in LTE, NR PHY Layer handles PUSCH (Physical Uplink Shared Channel) for data and PUCCH (Physical Uplink Control Channel) mainly for UCI (Uplink Control Information) transmission. It also supports PRACH (Physical Random Access Channel) for Random Access procedures, as in LTE, and SRS (Sounding Reference Signal) for channel estimation purposes. For PRACH and PUCCH different formats are supported, which have direct impact on the resources needed for transmit them. For SRS also different configurations are supported in terms of symbols used/PRB.

For UL and DL DMRS (Demodulation Reference Signals) are also supported. Typically the first PUSCH or PDSCH symbols are used for demodulation reference purposes.

As can be seen, given the variety of format and configuration supported for

all this Physical layer control channels and signals, is not easy to estimate a final rough overhead. For this thesis purpose, given the objectives of the developed simulator, it is assumed an overhead load based on the throughput estimation made in the section 4.1.2 of [31]:

- For FR1 (Frequency Range 1) that is, frequency bands bellow 7.125 GHz: 14% in DL and 8% in UL.

- For FR2 (Frequency Range 2) that is, frequency bands above 24.25 GHz: 18% in DL and 10% in UL.

2.3.6 Network Slicing

One may be wondering how network resources can be handled to support the different configurations needed for the different service's requirements. The answer is: using a feature called Network Slicing.

The basic idea behind this feature is to allocate network resources to different Network Slices, which act as virtual or logical networks with relative independence between each other, allowing to configure each one according to the service it will be used for. The Slice is defined end to end, so to support Network Slicing, RAN, Core and Transport Network must be prepared. At transport network SDN (Software Defined Network) will be an enabler for this. At Core level, NFV (Network Function Virtualization) is going to be used. In this way, each Core node will become a function mapped over certain physical resources defined in terms of compute capacity and storage over a data center, as can be seen in figure 2.20. Each Slice could have its specific network functions which data center resources will be allocated for. Resource allocation should be dynamically according to the service needs.

At RAN level, the resources to allocate to the different slices will be the spectrum, i. e. PRBs in the used band. Different Slices will have different configuration not just in terms of numerology, but also in terms of features supported as mobility and access random procedures for example, which will have a direct impact in the slice capacity and performance for the required traffic profile.

Note that the way in which resources are allocated to different slices is not specified by the standard, so is left to vendor implementation. The standard covers the involved new signalling, and some basic design criteria as slice isolation principles, but not the resource allocation algorithms between slices neither the way the services requirement are mapped to network configuration.

Also note that Network Slicing is not the only way of multiplexing different services in 5G Networks. The standard also considers the possibility of puncturing eMBB allocated resources for URLLC traffic (mini-slot allocation). However in this thesis only Network Slicing, and particularly, RAN Slicing is studied for different services coexistence.

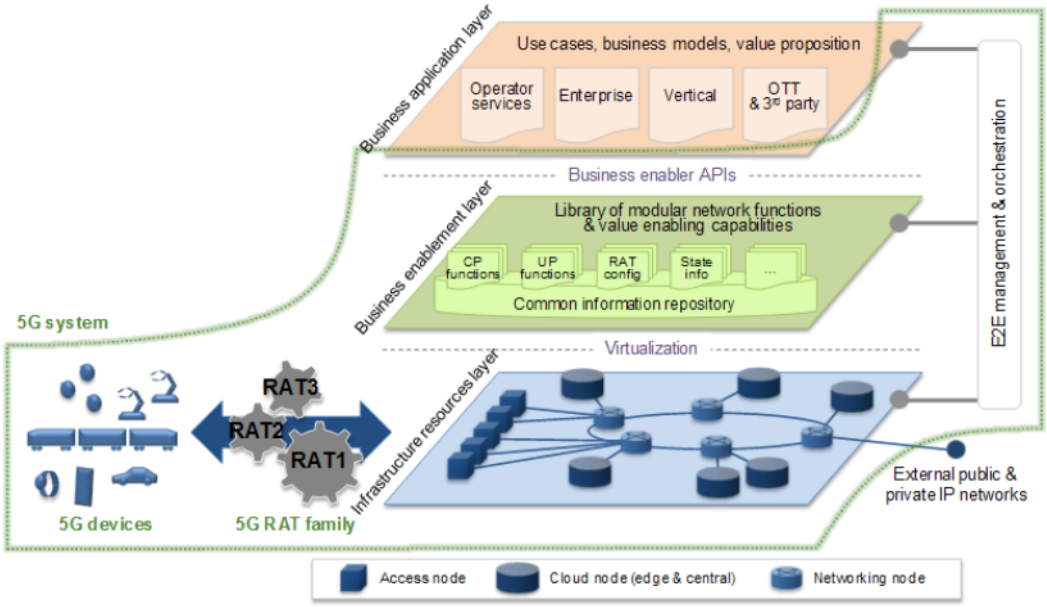

Figure 2.20: Network Slicing basic model from [3].

2.4 Chapter Summary

This chapter introduced the most important concepts for this thesis. NR key enablers and design concepts are explained, as a way to support service requirements evolution from LTE. The next chapters will cover the developed 5G simulator design, validation, and test cases results.

Chapter 3

Py5cheSim

This chapter makes a brief description of the developed simulator. First, the most important developing tools are presented followed by the simulator features. Then the simulator's architecture and basic operation are shown, followed by the most important design considerations made through the technology abstraction. Finally different scheduler implementations are introduced.

3.1 Development Tools

As it is been mentioned before, *Python* was used to develop the simulator. Python is a very powerful and versatile language, provided with tons of packages developed for specific purposes, from Discrete Event Simulation to Machine Learning tools. Although Python allows the use of different types of coding, this work uses OOP (Object Oriented Programming). Given the associated concepts and the relation between them, OOP was considered appropriated for this project, also thinking in future upgrades and extensions.

The tool used to implement Discrete Event Simulation was *SimPy* [40]. *SimPy* is a Python package that introduces an appropriate environment for discrete event simulations, solving the typical issues of implementing this type of simulators, as for example the event scheduler implementation. To execute actions in a "timed" way *SimPy* offers the Process class, and the PEM (Process Execution Method) methods. Basically any method that needs to be executed in a "timed" way must be a PEM method. This environment provides PEM activation mechanisms and an entire simulation framework to support the last.

Finally, code documentation was made using *pydoctor* [13].

3.2 Architecture and Basic Operation

Figure 3.1 shows some of the main concepts involved in a simple simulation. One cell may serve one ore more UE groups. Each UE group has a a number of UEs with a defined traffic profile. Traffic profile is described by the Packet Flow parameters. Packet Flow is transported through the air by Bearers. As *Py5cheSim*

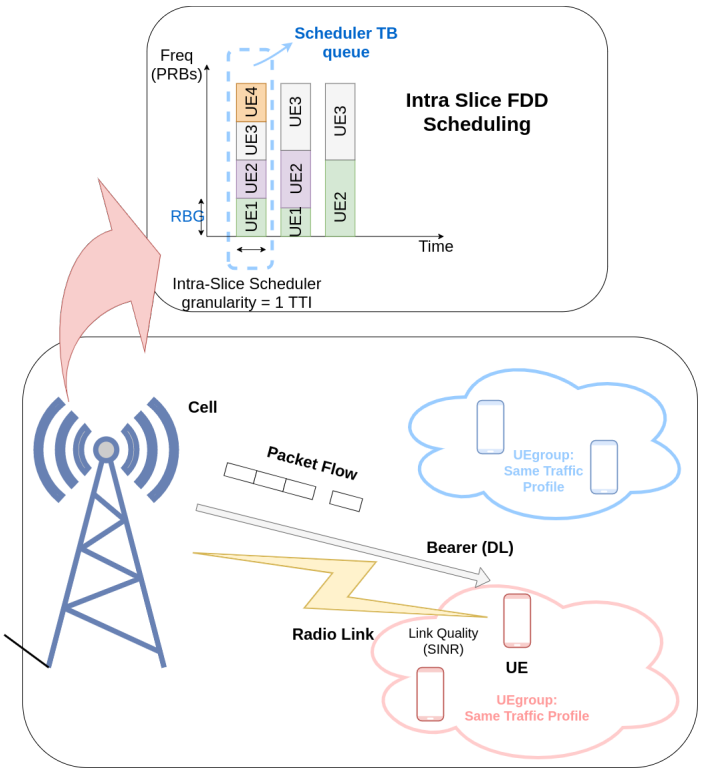

Figure 3.1: Py5cheSim main concepts general diagram.

focuses on the radio part of the network, only radio Bearers are considered. Each UE connected to the cell has a radio link with a quality given by the UE SINR. Packets are processed through a simplification of the radio interface protocol stack and transmitted over the air through Transport Blocks using resources from the time-frequency grid, as can be seen on the diagram.

Given the later considerations *Py5cheSim* is build on the next modules:

- *UE.py:* UE parameters and traffic generation.
- *Cell.py:* Cell configuration and statistics management.
- *Slice.py:* Slice configuration.
- *IntraSliceSch.py:* Base intra slice scheduler implementation.
- *InterSliceSch.py:* Base inter slice scheduler implementation.
- *Scheds_Intra.py:* Other intra slice schedulers implementation.
- *Scheds_Inter.py:* Other inter slice schedulers implementation.
- *simulation.py:* Is the simulation script. It configures and runs a simulation.
- *Results.py:* Provides auxiliary methods to present simulation results, and configure traffic profiles.

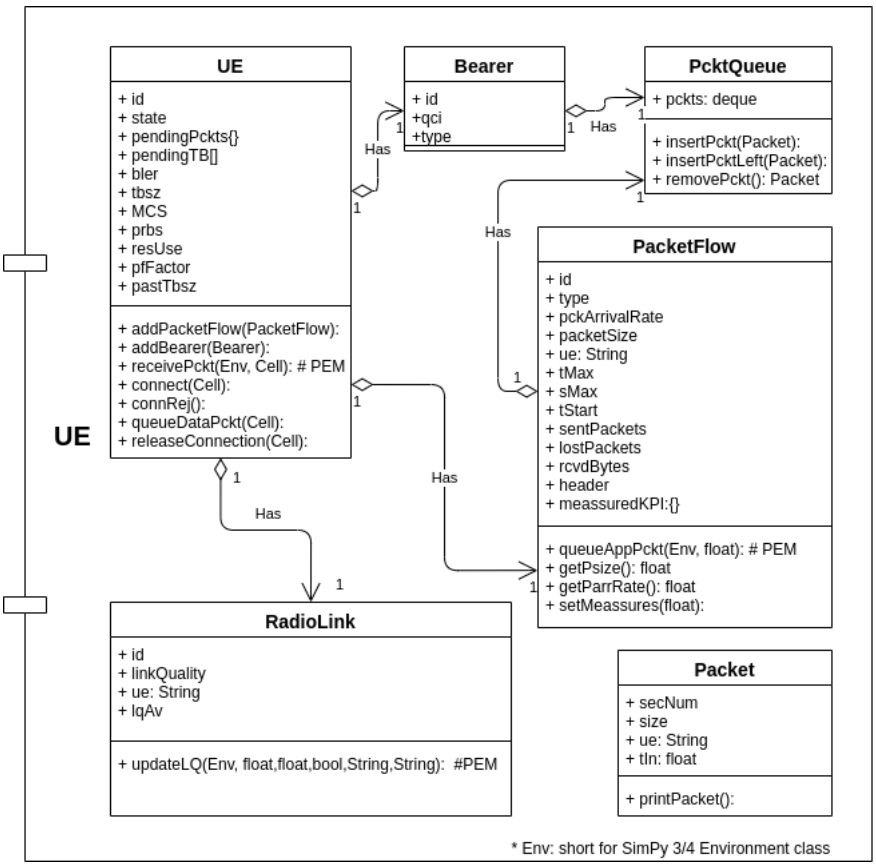

Figure 3.2: UE module class diagram.

The first five modules constitutes the simulator core, and contains several classes, as can be noted in figures 3.2 and 3.3. Particularly the modules *IntraSliceSch.py* and *InterSliceSch.py* have the basic schedulers for one and several slices respectively. The default algorithm in this classes is Round Robin. Other schedulers can be defined in the *Scheds_Intra.py* and *Scheds_Inter.py* as classes inherited from the Base Scheduler's ones defined in the former modules overwriting the *resAlloc* method. This was a design choice made to simplify new schedulers implementation. The main network model abstractions are contained in the simulator Core, so there is no need to have deep knowledge on that field to run a simulation or to integrate a new scheduler. In this way *Py5cheSim* provides a framework for 5G new scheduler algorithms implementation in a straightforward and intuitive way.

UE.py module contains all it has to do with UE properties and behaviour, bearers and traffic. Particularly, the *PacketFlow* class is responsible for creating different traffic flows, in terms of packet inter-arrival time and size, and collecting the following basic performance counters:

- sent packets
- received packets

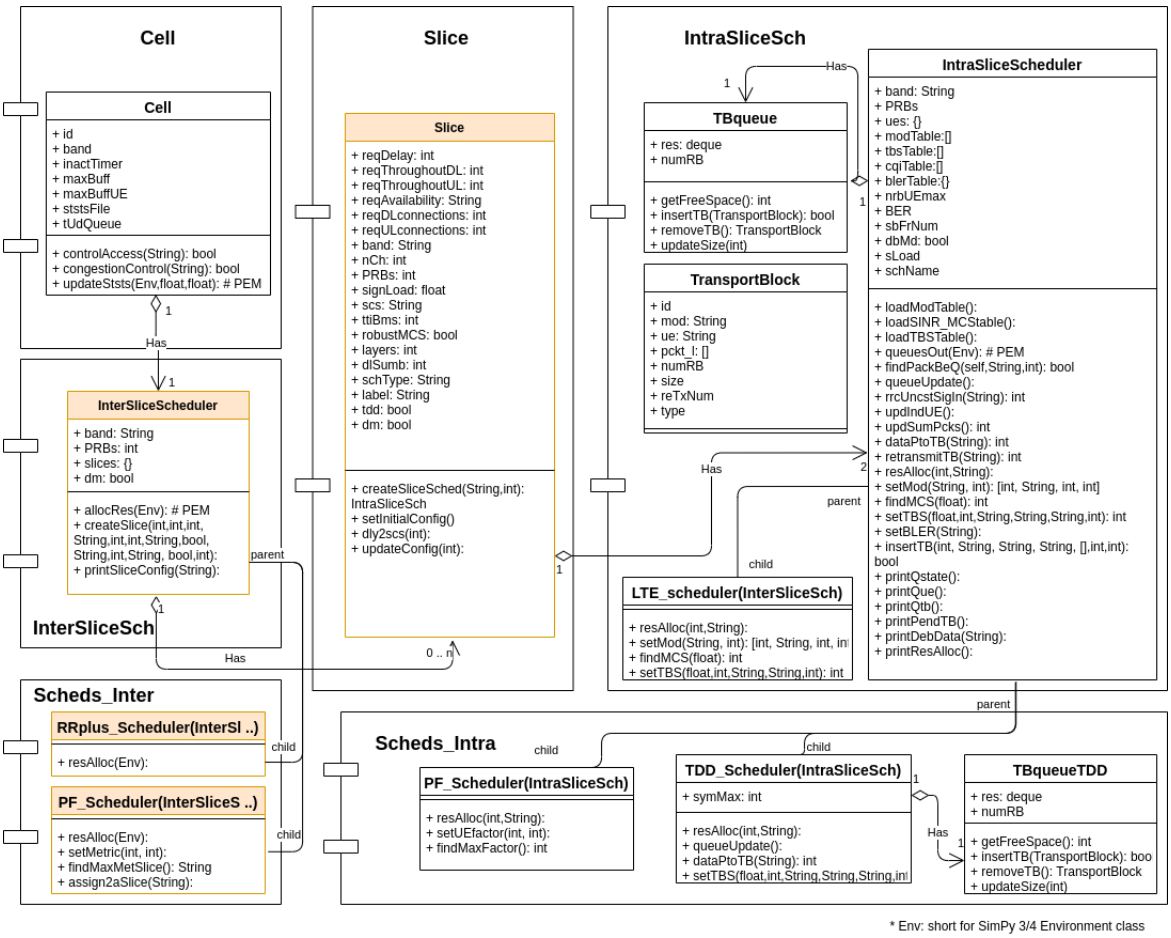

Figure 3.3: Cell, Slice and Schedulers modules class diagrams. The orange classes were developed for multi-Slice support.

- received bytes

Using this counters, the *setMeassures* method calculates the basic UE performance indicators considered by the moment:

- Packet Loss Rate (%)
- Throughput by user (Mbps)

Traffic profiles different from the described in subsection 3.2.8 could be defined by changing the way in which packets are generated, re-writing the *getPsize* and *getParrRate* methods from *PacketFlow* class.

Two classes were developed as a starting point to support inter Slice Scheduling support: *InterSliceScheduler* and *Slice* clases. The first one implements directly the basic inter slice scheduler, as it dynamically allocates band PRBs between the different configured Slices. The second one manages Slices requirements and translates to Slices configuration. For each Slice, two new instances of the *In-*

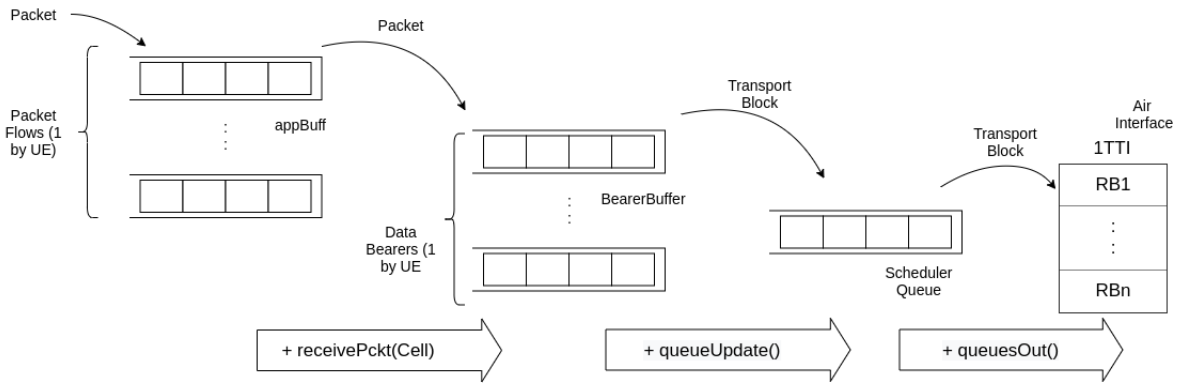

Figure 3.4: Queues operative[1].

traSliceScheduler class will be created (one for DL and another for UL scheduling) and will operate according to the slice configuration.

LTE_scheduler class inherits from *IntraSliceScheduler* class overwriting the *setMod* method to implement LTE scheduler considering the main differences with NR in terms of MCS selection, TBS calculation, and signalling overload. In this way LTE simulations can be executed creating one slice with the LTE scheduler.

The intra slice scheduler's basic operation can be seen in the figure 3.4. The application generates a packet flow through the *queueAppPckt* method. Each packet is stored in an application queue, the first who appears in the figure 3.4. Then, when the UE reach the RRC-connected state in the cell, the DRB is established and its packets go to the bearer queue, through the *receivePckt* method. Then, the scheduler assigns resources for all the active bearers, and takes packets from there to make TB with an appropriate MCS according the UE SINR at the moment, and put them in the Scheduler queue through the *queueUpdate* method. Finally the scheduler takes the TB from the queue at each TTI, and send them through the air interface. The TB are successfully received with a probability of (1-BLER).

The *simulation.py* file has no classes, because it was designed as a python script, to configure and run a simulation. It prints average simulation results in the terminal at the end of the simulation. The simulation file also triggers traffic statistics processing to make different kind of charts to show simulation results using the *Results.py* module. The last module contains several auxiliary methods for kpi processing and charts creation and a class to manage the UE's groups which the simulation runs for. It also contains a SINR generator, to define the initial SINR of each UE in the simulation.

3.3 Features Support

The following sections briefly describes *Py5cheSim* main supported features along with the considered models and design choices taken.

3.3.1 UL/DL Support

At the moment, the simulator supports only one UL or DL bearer by UE, and ACK in the opposite direction is not being considered. Furthermore, HARQ procedures are also not being considered to reduce the implementation complexity. The same MCS tables are applied for UL and DL, and overhead is introduced in TB size calculation according to throughput estimation in [31].

3.3.2 Different Traffic Profiles

The simulator supports different traffic profiles configuration by groups of UEs. Traffic profile is set in terms of average packet size (in bytes) and arrival rate (in ms). Packets of a size S will arrive at a rate R. S and R are random variables with Pareto distribution with a mean value equal to the packet size set for the simulation, and truncated to twice its value. This traffic model was based on the streaming traffic model considered in [11]. Different intensity traffic can be considered by setting S and R in a UE group, on the simulation script.

3.3.3 SINR Generation

As the main objective of *Py5cheSim* was to provide a tool for testing different scheduling techniques through cell capacity analysis, radio channel modeling was not a priority. In this first version, the core of the simulator basically takes UE SINR as an input. Cell load has no impact on UE SINR, because it is assumed that UEs activity will not have meaningful variation during the simulation. It is also assumed that UEs won't move during the simulation, so UE SINR will not have meaningful variation through the simulation. Besides, SINR values are assumed to be the same along the operating band, so there is no difference between different PRB for each user.

In the simulation file one can configure the initial SINR. The simulator supports two options for initial SNIR setting: all UEs using the same configured value, or each UE in the UE group having a different initial SINR value between 5 dB and a configured maximum. SINR then varies with time following a Gaussian distribution centered in the initial values with a small variance.

A possible future improvement for the developed simulation tool could be the integration of a channel model.

3.3.4 FDD/TDD Frame

Duplexing mode is set depending on the cell's band set for the simulation. MmWave bands use TDD, middle and low bands can use FDD or TDD. In this case duplexing mode can be chosen in the simulation file. Note that for FR1 bands the simulator does not check consistency between the chosen band and duplexing mode, neither between band and configured bandwidth. The reason is to allow simulation for more scenarios than the already standardized.

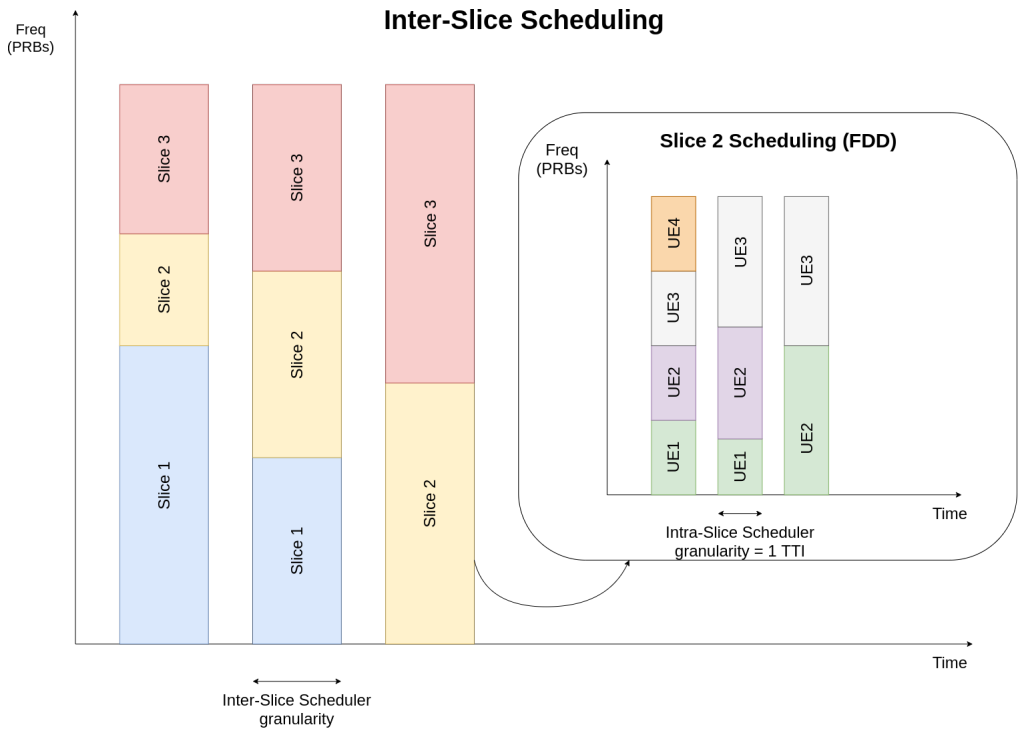

Figure 3.5: Inter and Intra Slice schedulers basic implementation scheme.

In FDD mode, users resource allocation is made in terms of Slice configured PRBs. This first *Py5cheSim* version supports only Resource allocation type 0 from [29]. In TDD mode, users resource allocation is made by slot. Although this version does not support mini-slot scheduling yet, the implemented TDD scheduler is prepared to support it by introducing a few changes in the *resAlloc* method.

3.3.5 Network Slicing

Py5cheSim provides a basic implementation of RAN Slicing as a core feature by the use of a two level scheduler composed of an intra Slice Scheduler and an inter Slice Scheduler. The first one is oriented to solve resource allocation between different UEs from the same slice, and the second one to allocate resources between the different slices instantiated on the cell. At inter slice level, resources are the available the cell's band PRBs. Figure 3.5 shows the basic idea behind this primary RAN Slicing implementation.

Each Slice has a a set of requirements and a configuration. Configuration is set automatically depending on Slice requirements in terms of delay, band, traffic load, UE capabilities and availability. For each Slice, numerology/SCS/TTI, duplexing mode, scheduler algorithm to use and signalling load are set at the instantiating moment, in the *setInitialConfig* method from *Slice* class. More detail can be added in the implemented mapping by modifying this method. In this first *Py5cheSim* version intra and inter slice scheduler algorithms are indicated as an input in

the simulation script. Allocated Slice PRB can change dynamically according to inter Slice scheduler decision with a configurable time granularity. In this first version only most commonly used traditional schedulers are implemented, but new scheduler algorithms can be easily added by inheriting from the basic Round Robin schedulers and overwriting the methods involved in the resource allocation process (*resAlloc*).

3.3.6 Multiple Numerology Support

Each Slice supports any of the numerologies shown in Figure 2.16. Along with the SCS, TTI and slot duration are set in terms of handled slots/ms. At the moment, Slice numerology is set according to the required Delay. Table 3.1 shows the configured mapping between RAN Delay requirements and SCS configuration for a Slice. Note that the considered thresholds were defined taking into account the delay analysis and results presented in [27] assuming average values for the scheduling timings, and the direct dependence with the slot duration and SCS as a consequence.

Delay Requirement (ms)	≤ 2.5	≤ 5	≤ 10	> 10
SCS (kHz)	120	60	30	15

Table 3.1: Delay requirement to SCS implemented mapping.

3.3.7 256QAM

3GPP TS 38.214 table 2 (table 5.1.3.1-2), supporting 256QAM, is used to set modulation order (Qm) and Code Rate (R), for UL and DL. In real systems, different MCS tables are used for UL and DL, but in this first *Py5cheSim* version only DL MCS table 2 is used in both directions to provide results that can be compared with the obtained with *5G-LENA*. That is because *5G-LENA* only adds DL MCS tables 1 and 2 from 3GPP TS 38.214 [29], as can be seen in the files *nr-eesm-t1* and *nr-eesm-t2* from the module source. However, new MCS tables for UL can be easily added in *Py5cheSim* by means of simple modifications in the basic intra slice scheduler.

3.3.8 Carrier Aggregation

A the moment, the simulator supports aggregation of carriers as defined in the configuration file, with no load balancing between them. Aggregated carriers resources are available to TB allocation under the same cell object. It simply works as an increase of the cell bandwidth even though in real networks different cells carriers can be aggregated and a UE can take their resources according to a load balancing policy configured and its capabilities.

3.3.9 SU/MU-MIMO

The use of SU/MU-MIMO can be configured in the simulation file, along with the number of layers considered. In the case of SU-MIMO, the allocated resources by UE will be multiplied by the number of layers and consequently the UE throughput. In the case of MU-MIMO, even though there will be also more resources, the allocated RB by UE is maintained, so more UEs can be solved in the same TTI. In both cases it is assumed that all UEs in the group and the cell support the configured scheme. In real networks MIMO use will depend on radio conditions measured in terms of Rank Indication, and UE and network capabilities.

3.3.10 Different Scheduler Implementations

By default both IntraSlice as InterSlice schedulers implements Round Robin algorithm. However, Proportional Fair Scheduler is also implemented at the moment, and other scheduling algorithms can be added developing an *intraSliceSched* child class. The IntraSlice and InterSlice scheduler algorithms can be set in the configuration file.

3.4 Design Considerations

This section presents the main abstractions made from the standard through the implemented model in the simulator for both 5G and LTE schedulers.

3.4.1 5G Scheduling Modeling

In the case of the 5G basic intra slice scheduler, the following design considerations apply:

- The MCS allocation is based purely on UE's SINR. SINR-MCS tables was generated running the *cttc-nr-demo.cc* example from *5G-LENA* for a wide range of SINRs.

  In *cttc-nr-demo.cc* example BLER model is set as ns3::NrEesmCcT2, and AMC Model as Nr::ErrorModel. This means that MCS allocation is made dynamically to keep BLER under 10%, and MCS table 2 from 3GPP TS 38.214 [29] will be used to include 256QAM modulation. More details about *cttc-nr-demo.cc* example can be found in the section 4.4.1 (next chapter).

- Channel coding rate (R) and number of bits by symbol (Qm) allocation is made using table 5.1.3.1-2 from 3GPP TS 38.214 [29], assuming the use of 256QAM is possible for UL and DL. Note that UL MCS tables actually does not include 256QAM. Adding a different MCS table for UL may be considered for future improvements, and can be done easily as it is been said in the previous section.

- On the TBS calculation, the procedure referenced in 3GPP TS 38.214 [29] was implemented, with some assumptions. First, TBS is estimated as N_{info} in 3GPP TS 38.214 [29], to reduce complexity and have results that could be compared to *5G-LENA*'s, as can be seen in 3.1.

$$TBS = N_{\text{info}} = N_{\text{RE}} \times R \times Qm \times l \tag{3.1}$$

  Here l is the number of layers used in case of SU-MIMO. R and Qm are obtained from the MCS table as mentioned before. N_{RE} is given by equation 3.2 where n_{PRB} is the number of PRB allocated to the UE. $\overline{N'}_{\text{RE}}$ is estimated as can be seen in equation 3.3 to avoid adding more tables.

$$N_{\text{RE}} = \bar{N'}_{\text{RE}} \times n_{\text{PRB}} \tag{3.2}$$

$$\bar{N'}_{\text{RE}} = min(156, 12 \times SlotSymbols \times (1 - OH)) \tag{3.3}$$

  Here $\overline{N'}_{\text{RE}}$ is representing the number of resource elements used for PDSCH or PUSCH. It is assumed that the number of overhead symbols in a PRB (*12* x *SlotSymbols* x *OH*) responds to the established in the throughput calculation (point 4.1.2) of 3GPP TS 38.306 [31]. That is, in FR1 *OH* is configured as 0.14 for DL and 0.08 for UL. In FR2 *OH* is configured as 0.18 for DL and 0.10 for UL. It is important to note that overheads considered later are including all possible resource use for other things different from the PDSCH and PUSCH (SSB, PHY channels and signals). In real networks this channels and signals will imply a variable overhead, but in this *Py5cheSim* version the later assumption was made to simplify the implementation. Future versions could have a more realistic approach.

- BLER percentage is based on the results obtained from the *5G-LENA* example. As in test simulations made for validation purposes the resulting BLER was zero in all cases for the obtained SINR-MCS tables, at the moment in *Py5cheSim* BLER is set to zero. However, the possibility to re transmit a TB is considered in the implementation just changing the *BLER* attribute value through the *setBLER* method in *intraSliceScheduler* class.

- The resource allocation scheme follows the standard's approach: in frequency resource allocation type 0 is used. The RBG size depends on the band size in use as stated in tables 5.1.2.2.1-1 and 6.1.2.2.1-1 in 3GPP TS 38.214 [29]. It is also possible to allocate the entire band to each UE in a TTI basis. Resource allocation to a UE is made when there are available resources for that, not considering the scheduling timing parameters given by the standard (K0, K1, K2), again, for simplification purposes. Future *Py5cheSim* versions with delay measurements support should consider it.

- In time domain, different slot types are supported when using TDD by changing the number of DL symbols indication in one slot. Resource allocation is made here with a slot granularity by default, and it is assumed that different UEs will be using different beams and the use of beamforming has

no impact on resource allocation other than the estimated with overheads later considered. As a consequence, resource allocation in a TDD cell will be only in time domain. Initial DL/UL symbol allocation is defined according to the relation in traffic profiles and number or UEs configured in each direction. DL/UL symbol allocation is static in this version of *Py5cheSim*. Dynamic DL/UL symbol allocation can be implemented in a new inter slice scheduler simply building a new class which inherits from *interSliceScheduler* and overwriting the *resAlloc* method.

- Packets in Bearer buffer can be fragmented and concatenated to fit in the TBS.

3.4.2 LTE Scheduling Modeling

In the case of the LTE basic scheduler, the following design considerations apply:

- Only DL was considered for the LTE scheduler because LTE scheduler was developed at the begining of the project, only for basic structure validation purposes (*5G-LENA* simulator wasn't available yet).
- Spectral efficiency was calculated using the Shannon Law equation, configured BER and UE SINR as can be seen in equation 3.4.

$$SpecEff = log_2\left(1 - \frac{SINR}{\frac{\ln 5 \times BER}{1.5}}\right) \tag{3.4}$$

  SINR is generated for each UE using the SINR generator in the *results.py* module, and BER is set in the *LTE_scheduler* class.

- CQI is set according to the spectral efficiency calculated before using the *cqiTable* (table 7.2.3-1 from TS 36.213 [28]).
- Using the quantized Spectral Efficiency from the *cqiTable*, MCS and TBS (Transport Block Size) index is determined from the table *modTable* through the procedure and table presented in R1-081483 (as in the *lena* module from *ns3*) and table 7.1.7.1-1 from TS 36.213 [28]. From the MCS table presented in R1-081483 MCS index is obtained for the spectral efficiency, and table 7.1.7.1-1 from TS 36.213 shows the mapping between MCS index and TBS index. The *modTable* in *Py5cheSim* contains the resulting mapping between spectral efficiency and TBS index.
- Using the TBS index TBS is obtained from the *tbsTable* (table 7.1.7.2.1-1 from TS 36.213 [28]).
- The amount of PRB allocated to each UE is set in the *resAlloc* method. PRB are allocated by RBG, which depends on the bandwidth available, as specified in Table 7.1.6.1-1 from TS 36.213. Resource Allocation Type 0 was always assumed for PDSCH.

- Packets in Bearer buffer can be fragmented and concatenated to fit in the TBS.
- BLER is determined by user using the *blerTable* depending on SINR, MCS allocated, and BER configured. *blerTable* is based on results obtained runing the *lena* module for the same input.

3.5 Schedulers

Two possible scheduling algorithms for intra and inter slice scheduling are actually supported by *Py5cheSim*:

- Round Robin
- Proportional Fair, with configurable exponents

In the following, intra and inter slice implementation for each one are briefly described.

3.5.1 Intra Slice Schedulers

For FDD simulations resource allocation for different UEs here is done in terms of PRB. The next scheduling algorithms are supported.

- **Round Robin**: the same amount of resources are allocated to the different users with active bearers and non empty buffers, with no preference between each other. This is the default scheduler implemented in the *LTE_scheduler* and *intraSliceScheduler* classes. Note that even if there is no preference, users with lower SINR tend to use more resources than users with higher SINR, because the former ones fill the bearer buffer with packets more often than the last ones.
- **Proportional Fair**, with configurable exponents: cell resources in each TTI are allocated to the user who has the biggest relation between actual achievable throughput, and past throughput:

$$\frac{actualThroughput^{\text{numExp}}}{pastThroughput^{\text{denExp}}} \tag{3.5}$$

  There are different ways to calculate the past throughput in the current bibliography. In this implementation, the past throughput is the average between the throughput reached during a configurable number of subframes through the *promLen* attribute of *PF_Scheduler* class.

  Note that as different exponents can be configured for numerator and denominator in the metric expression (ec. 2.1), different schedulers can be obtained. For example, configuring numExp = 1, and denExp = 0, the results of a Maximum Rate Scheduler can be obtained.

For TDD simulations resource allocation is done in a TTI granularity along the entire band. Only Round Robin TDD scheduler is supported at the moment. Other algorithms can be added as new classes inherited from *TDD_Scheduler* class.

3.5.2 Inter Slice Schedulers

The inter Slice Scheduler allocates resources for each slice with a configured time granularity always longer than the maximum possible TTI. The resources are the sub-band assigned for each slice, in terms of PRB (Physical Resource Blocks). The maximum TTI is the one obtained from the minimum subcarrier spacing configuration, which is 1 ms, corresponding to 15 kHz. For the inter Slice Scheduler, at the moment the following algorithms are supported:

- **Round Robin**: allocates statically the same amount of PRB to each defined slice, independently of the bearer buffer size in each slice. It is implemented in *interSliceScheduler* class as the default inter Slice Scheduler. New algorithms can be implemented by extending this class in the *Scheds_Inter module*.
- **Round Robin Plus**: allocates the same amount of resources to each slice with packets in UE bearer buffers. If a slice has no packets in any of their UE's bearers, it will not be considered for resource allocation and resources will be divided between the other slices.
- **Proportional Fair**, with configurable exponents: allocates cell resources based on the relation between current possible Slice throughput in terms of average UE possible TBS, and past Slice throughput, in terms of received bytes, as can be seen in the next equation:

$$\frac{averageUeTBS^{\text{numExp}}}{receivedBytes^{\text{denExp}}} \tag{3.6}$$

The simulator allows to configure the time granularity for the InterSlice scheduling decision by setting the *granularity* attribute in the *interSliceScheduler* class. It also allows to configure the duration (in seconds) of the received bytes history to consider in the metric calculation, by setting the *rcvdBytesLen* in *PF_scheduler* class.

3.6 Chapter Summary

In this chapter *Py5cheSim* features, architecture and main design considerations were presented and explained. The next chapter will show the tool validation tests results, and possible use cases, followed by this thesis conclusion.

This page has been intentionally left in blank

Chapter 4

Simulator Validation

This chapter presents *Py5cheSim* validation methodology and its main results. The validation was made through the *nr* module from *ns3* simulator, also known as *5G-LENA* [8], and theoretic throughput calculator tool from [42]. The general idea behind this validation was to test the developed simulator operation and compare performance results with the references in terms of the main KPI considered. The goal is to not exceed 10% of error margin. Note that the developed simulator implies a really big simplification of the standard procedures, so an error margin should be expected.

Simulator validation was made in two levels: An intra-Slice validation level, and an inter-Slice validation level. In the first case, validation was made in terms of:

- AMC operation and resulting TBS
- Throughput measures
- Different implemented intra-Slice schedulers operation
- Features support: CA and MIMO

In the second case, validation was made in terms of:

- Slice Management
- Different inter-Slice schedulers operation

4.1 Intra-Slice level Validation

This section describes Intra-Slice level validation tests and shows comparison results with those obtained through the reference tools.

4.1.1 Reference Validation Tools

Most of Intra-Slice validation was made by comparing *Py5cheSim* simulation results with those obtained with the *cttc-nr-demo.cc* example from *5G-LENA* simulator. The *nr* module from *ns3* (also known as *5G-LENA*) was chosen as a reference tool for validation given the extensive testing and usage of its predecessor, the *lena* module from *ns3*, which was taken as a baseline for the *nr* module development. The *lena* module is available since 2011 and has evolved along with the standard adding most of the LTE features released by 3GPP. It has been used and tested in a significant amount of research works about LTE, including the LTEst project [11].

The *cttc-nr-demo.cc* contains a simulation basic example with one gNB and a configurable number of UEs downloading packets with a configurable traffic profile. Cell's band, bandwidth and numerology are configurable. Although the simulation has configured two bandwidth parts, the original script was adapted to use only one. Also a few lines were added to support UL traffic, configure Error based AMC model, MCS table 2 from 3GPP TS 38.214 and the scheduling algorithms to test. Different SINR were generated by changing UE distance to gNB, or the UE/gNB transmission power. Also, for schedulers validation in FDD a few changes in *grid-scenario* helper was introduced to have more than one UE with different SINR.

A Python script was created to run simulations in *5G-LENA* and then in *Py5cheSim*, with the same scenario, traffic profile and configuration. This script parses the resulting traces in the *RxPacketTrace.txt* file from *5G-LENA* and the results obtained from *Py5cheSim* and build charts with the main kpi considered for validation, for each SINR. Note that, although in *Py5cheSim* traffic profile has a random component, as for validation purposes only full buffer traffic profile was considered, it wasn't found necessary to make an statistical approach for that. *Py5cheSim* and *5G-LENA* results are taken as deterministic in this validation procedure.

As for the theoretic throughput calculation tool from [42], it basically consist in the application of the equation for maximum UE throughput from [31]. Features validation was made using this tool configuring cell bandwidth, numerology, number of carriers or layers depending on the case, R and Qm. In TDD case was also configured DL/UL proportion of symbols in one slot. Again, as full buffer traffic model was used, *Py5cheSim* results were taken as deterministic.

4.1.2 NR AMC and TBS Validation

AMC validation was made comparing MCS allocation with *cttc-nr-demo.cc* script results for a wide range of SINRs, using the same band and bandwidth, no CA nor MIMO, and only one UE with full buffer DL traffic profile. Figure 4.1 shows validation results. Similar results can be obtained for full buffer UL traffic.

TBS validation was made using the same script to compare MCS. TBS comparative results can be seen en Figure 4.2.

As can be seen, MCS are allocated mainly according to *5G-LENA* results,

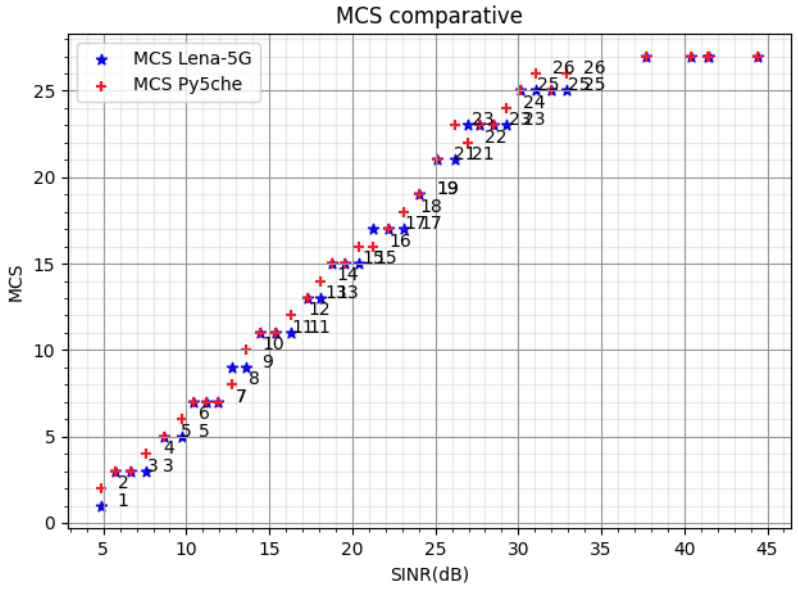

Figure 4.1: Py5scheSim vs 5G-LENA MCS comparison example for a 5 MHz FDD cell with DL full buffer traffic.

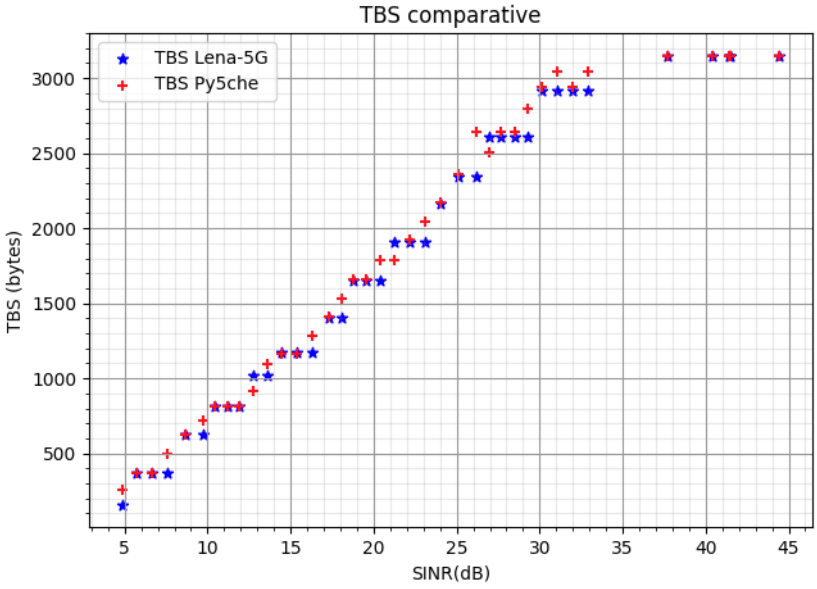

Figure 4.2: Py5scheSim vs 5G-LENA TBS example.

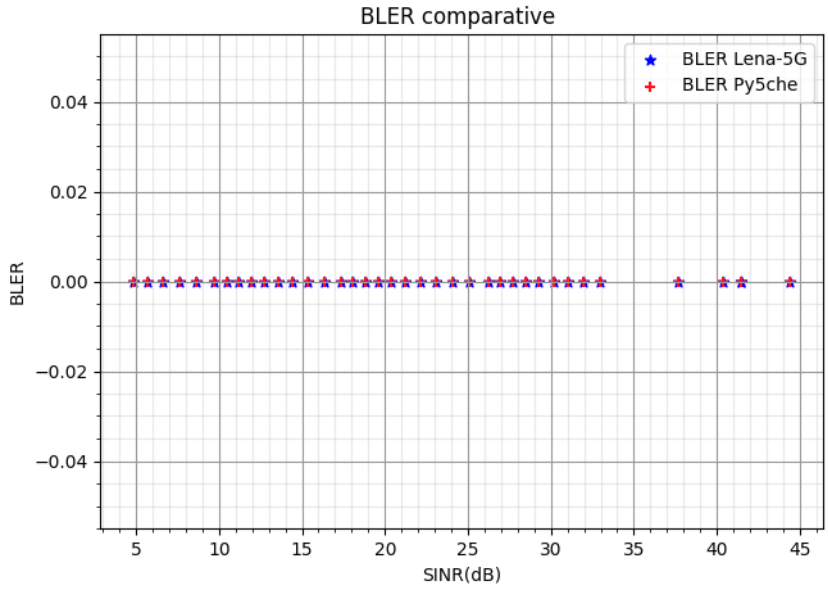

Figure 4.3: Py5cheSim vs *5G-LENA* BLER example.

as expected considering how AMC was implemented in the developed simulator. Remember that *Py5cheSim*'s MCS tables were defined according to *5G-LENA*'s simulation results configuring AMC based on Error model as a reference. AMC algorithm is implementation dependent and often configurable in real systems, so its evaluation is beyond the scope of this thesis. More realistic AMC algorithm implementation can be done overwriting the *setMod* method in the scheduler implementation. The same occurs with BLER. *Py5cheSim*'s BLER was adjusted to *5G-LENA*'s simulation results, but more realistic BLER model can be implemented overwriting the *setBLER* method.

Py5cheSim MCS allocation was adjusted to the results obtained with *5G-LENA* to be able to use *5G-LENA*'s results as a reference to compare for validation purposes. As UE and cell throughput will strongly depend on MCS allocation, throughput comparison could not be possible if the different simulation tools allocate different MCS. With that said, the few differences noted between *Py5cheSim* MCS allocation and *5G-LENA*'s respond to the simplifications made in the developed simulator along with the finite tabular results management.

TBS comparative results are also according to expected, considering the implementation, and the dependency with the AMC latter described. Note that overhead parameters in TBS formulas were adjusted to *5G-LENA* simulation results only for validation purposes.

BLER comparative results shown in Figure 4.3 are also according to expected, considering the implementation. Note that results obtained in Figure 4.3 for *5G-LENA* could be explained considering the BLER allocation mechanism in *5G-LENA* and the resulting MCS shown in Figure 4.1. Given the error model and AMC option configured, *5G-LENA* allocates the highest MCS with BLER lower

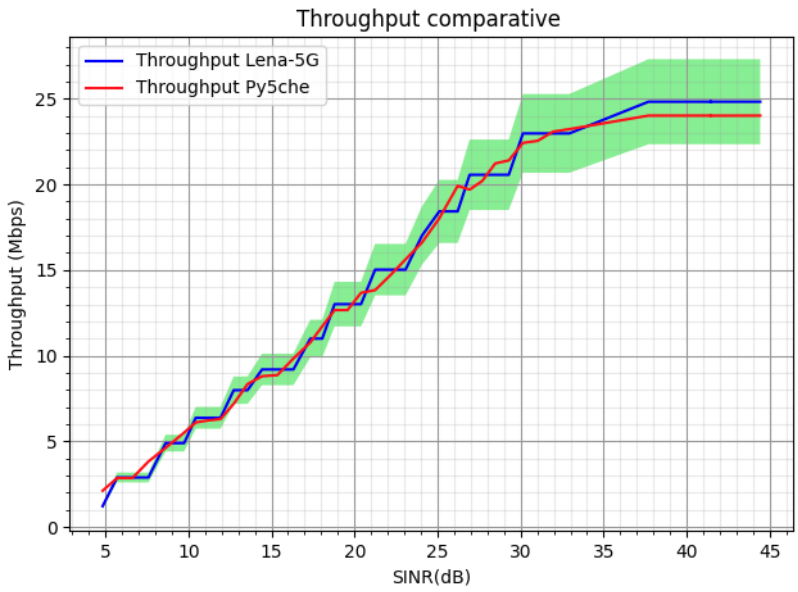

Figure 4.4: Py5cheSim vs 5G-LENA Throughput example.

than 0.1. As BLER values are tabulated it is possible that for the considered code word size and SINR, BLER values higher than 0 were not lower than 0.1. Further more, Figure 4.1 shows that MCS 27 is only allocated for SINR higher than 35 dB and for SINR between 20 and 30 dB (which are generally considered as good radio condition) allocated MCS are between 15 and 24. This restrictive MCS allocation is reasonable in a network with no margin for error in the air interface. So although the observed values may seem strange for the configured AMC and error model, resulting MCS are consistent with a BLER equal to zero.

4.1.3 Throughput

Throughput validation was made in two ways: comparing the former script results with *Py5cheSim*'s, and comparing the latter with the web throughput calculation tool's, on the same scenario. Figure 4.4 shows *5G-LENA* vs *Py5cheSim* results.

Throughput results are strongly influenced by MCS allocation and TBS calculation methods, so differences in MCS produce changes in the resulting throughput. Note also that TBS differences with *5G-LENA* can be expected, because *Py5cheSim* considers a fixed overhead while in *5G-LENA* it can change according to DMRS allocated symbols. However, even with the noted differences in TBS calculation methods, throughput results does not present meaningful differences.

In Tables 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6 maximum throughput comparison with the the tool available on [42] is shown. Throughput is calculated considering the highest MCS in use, and no CA nor MIMO, for different bandwidths and SCS, covering FDD and TDD cases. Error is calculated as the difference between results relative to Throughput calculator's value.

When comparing *Py5cheSim*'s results with those obtained from the theoretic

Bandwidth	Py5cheSim	Throughput Calculator	Relative error
5 MHz	25.3 Mbps	26 Mbps	3%
10 MHz	52.8 Mbps	56 Mbps	6%
20 MHz	106.7 Mbps	114 Mbps	6%

Table 4.1: DL FDD Throughput comparison for 15 kHz SCS

Bandwidth	Py5cheSim	Throughput Calculator	Relative error
5 MHz	27.2 Mbps	28 Mbps	3%
10 MHz	56.5 Mbps	60 Mbps	6%
20 MHz	112.6 Mbps	122 Mbps	8%

Table 4.2: UL FDD Throughput comparison for 15 kHz SCS

throughput calculation tool, most differences are explained by the TBS calculation method. Throughput calculator on [42] does not trunk N'_{RE} value.

Bandwidth	Py5cheSim	Throughput Calculator	Relative error
5 MHz	22.4 Mbps	24 Mbps	7%
10 MHz	48.8 Mbps	52 Mbps	6%
20 MHz	102.6 Mbps	110 Mbps	7%

Table 4.3: DL FDD Throughput comparison for 30 kHz SCS

Bandwidth	Py5cheSim	Throughput Calculator	Relative error
5 MHz	23.9 Mbps	26 Mbps	8%
10 MHz	51 Mbps	54 Mbps	6%
20 MHz	107.9 Mbps	116 Mbps	5%

Table 4.4: UL FDD Throughput comparison for 30 kHz SCS

Bandwidth	Py5cheSim	Throughput Calculator	Relative error
50 MHz	255 Mbps	270 Mbps	6%
100 MHz	509 Mbps	538 Mbps	5%
200 MHz	999 Mbps	1078 Mbps	7%

Table 4.5: DL TDD Throughput comparison for 60 kHz SCS

Bandwidth	Py5cheSim	Throughput Calculator	Relative error
50 MHz	282 Mbps	296 Mbps	4%
100 MHz	559 Mbps	592 Mbps	6%
200 MHz	1093 Mbps	1182 Mbps	8%

Table 4.6: UL TDD Throughput comparison for 60 kHz SCS

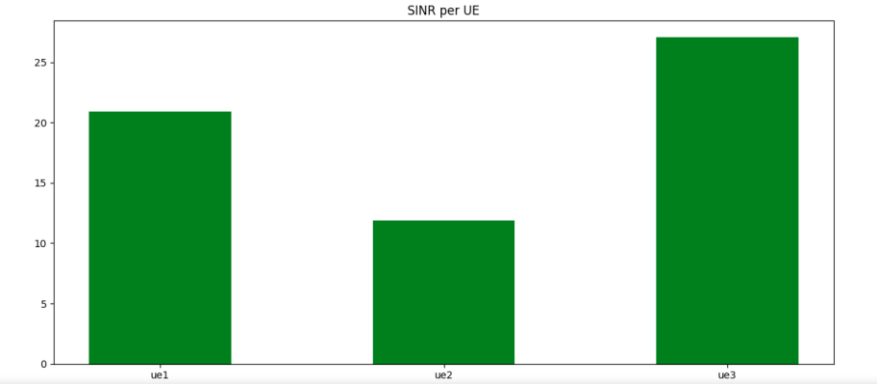

Figure 4.5: SINR per UE for the three UEs considered in scheduler validation tests.

4.1.4 Schedulers

The following sections present scheduling validation results. Scheduling validation was made comparing *5G-LENA* script results with *Py5cheSim*'s on a scenario with 3 UEs with full buffer traffic profile and different SINRs, using Round Robin and Proportional Fair schedulers. Figure 4.5 shows SINR per UE. The comparison script in this case use the SINR values obtained from *5G-LENA* for the three users as an input for *Py5cheSim*, assuring the same SINR values are used in the two simulation tools.

Note that even tough resource allocation type 0 was implemented in the developed simulator for the FDD case, there is a difference respect to the standard procedure and the *5G-LENA* implementation. In *Py5cheSim* if more than one RGB is allocated to a user, allocated RGBs to the same UE in the same TTI will not be consecutive. Although the later implies a difference in TBS results, the number of PRB allocated by TTI per UE remains unchanged, as can be seen in the following subsections. UE throughput also has no impact because of this.

Round Robin

Figure 4.6 shows Round Robin resource allocation per user. As can be seen, *Py5cheSim* distributes PRB between the three users equally, as expected from the Round Robin algorithm implementation. Differences with *5G-LENA* can be explained considering the implementation differences latter noted.

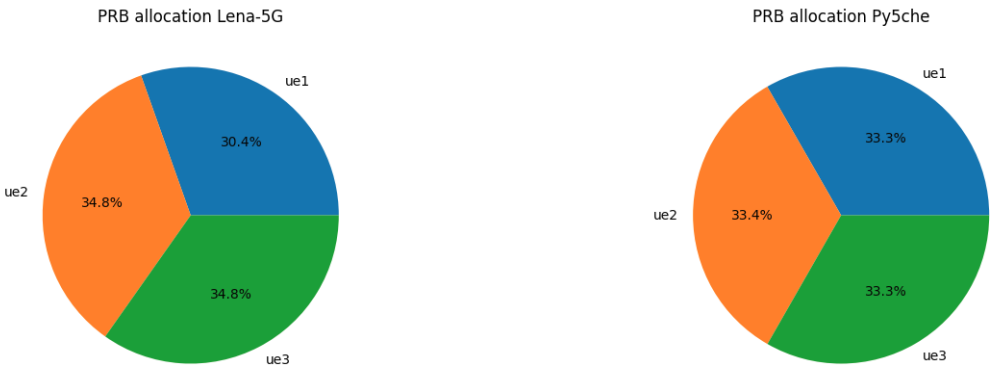

Figure 4.6: PRB allocation distribution per UE using Round Robin scheduler.

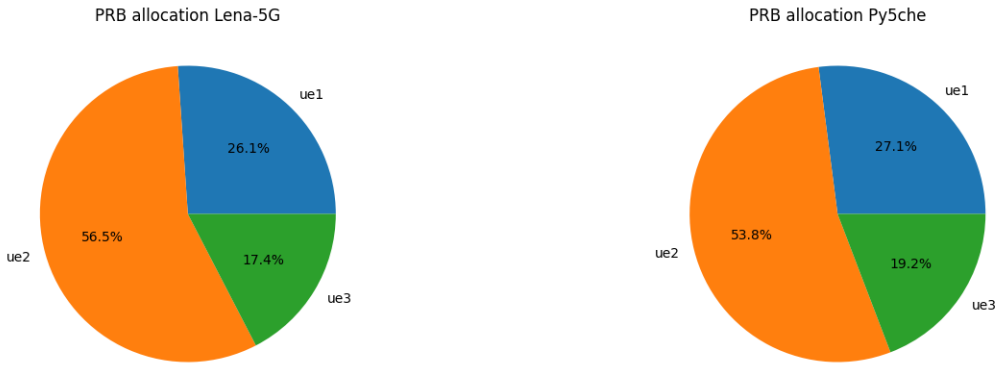

Figure 4.7: PF01 PRB allocation distribution per UE.

Proportional Fair

In this case three variants of the algorithm were tested, in the conditions explained before.

- Proportional Fair with numExp = 0 , and denExp = 1, PF01 from here on.
- Proportional Fair with numExp = 1 , and denExp = 1, PF11 from here on.
- Proportional Fair with numExp = 1 , and denExp = 0, PF10 from here on. In this case, as *5G-LENA* does not allow to configure the past throughput exponent and the possible throughput exponent (alpha) value must be between zero and one there is not comparison, showing the results obtained with *Py5cheSim*.

Figures 4.7, 4.8, and 4.9 shows PF01, PF11 and PF10 resource allocation per user.

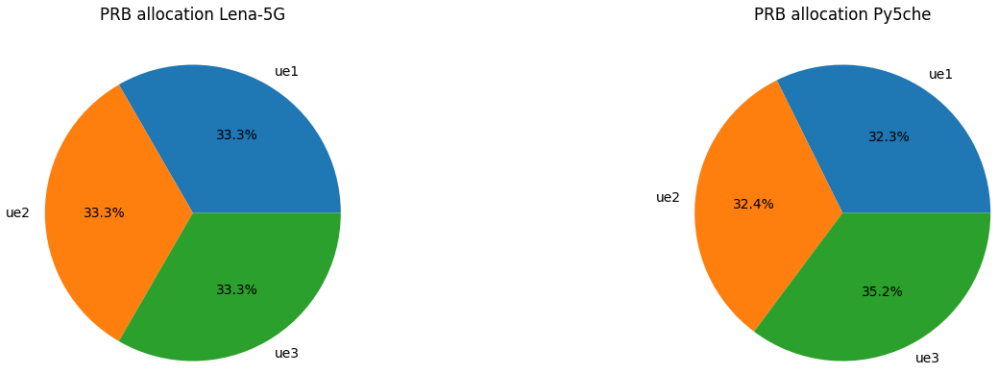

Figure 4.8: PF11 PRB allocation distribution per UE.

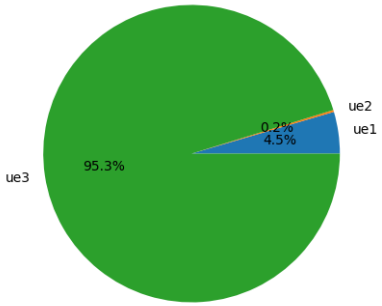

Figure 4.9: PF10 PRB allocation distribution per UE for three UE *Py5cheSim* simulation.

When using PF01 most resources are allocated to UE2, and less to UE3, as expected considering UE2 has the lowest SINR and UE3 the higher, and PF01 scheduler gives resources to the user with lowest past throughput. When using PF11 resources are allocated almost evenly between users as expected, considering that FP11 scheduler takes into account both, possible throughput for the radio conditions and past throughput so a good balance in resource allocation should be expected. When using PF10 most resources are allocated to UE3, and less to UE2, as expected considering UE2 has the lowest SINR and UE3 the higher, and PF10 scheduler gives resources to the user with biggest possible throughput, i.e. the biggest SINR.

TDD

TDD implemented scheduler assumes that all UEs are served through different beams, so resource allocation is made only in time domain. Although by default cell resources are assigned with a slot granularity, the *TDD_scheduler* class allows mini-slot allocation by modifying the *resAlloc* method. At the moment, only

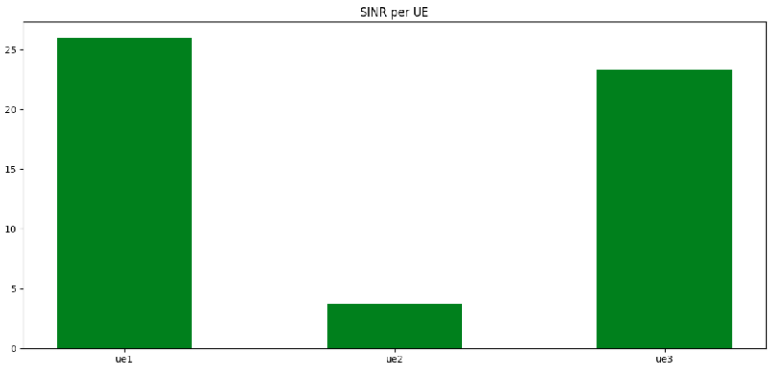

Figure 4.10: SINR per UE in TDD scheduler validation test.

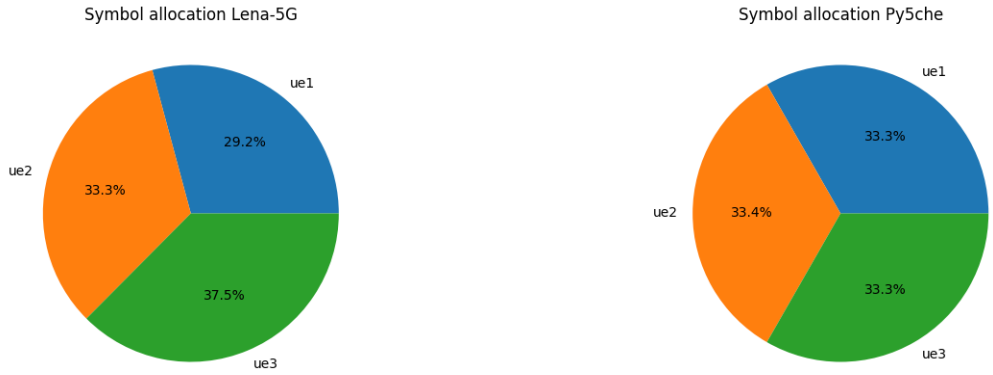

Figure 4.11: Symbol allocation distribution per UE

Round Robin algorithm is available.

TDD scheduler was validated with the same procedure described for the FDD case, only changing SINR by user, as can be seen in figure 4.10.

For this test, *resAlloc* method on *TDD_Scheduler* was modified to schedule UEs with a symbol granularity, to be comparable with resource allocation made in *5G-LENA*. Note that, as in the FDD case, if more than a symbol are allocated to a UE, it will not be consecutive. Again, although it implies a difference in TBS results, the number of symbols allocated by TTI per UE remains unchanged, as can be seen in the figure 4.11.

As can be seen, *Py5cheSim* distributes symbols between the three users equally, as expected from the Round Robin algorithm implementation. Differences with *5G-LENA* can be explained considering the implementation difference latter noted.

In TDD case, UL/DL scheduling was also validated considering two UE groups, one with UL traffic and other with DL, simultaneously. The number of symbols for DL and UL is configured as an attribute in the *Slice* class. In this case, for DL eigth

symbols was used, while four was used for UL. UE throughput was compared with the obtained with *5G-LENA* using the same amount of symbols in each direction. Table 4.7 shows UE throughput comparison for each case.

Direction	5G-LENA	Py5cheSim	Relative error
DL	152 Mbps	144 Mbps	5%
UL	68 Mbps	64 Mbps	6%

Table 4.7: TDD Throughput comparison using eigth symbols for DL and four for UL

The presented results show that the different implemented schedulers on *Py5cheSim* operates according to the expected for each algorithm in terms of resource allocation distribution between users.

4.1.5 Features

In the following, MIMO and CA features validation results are presented. Both features validation were made comparing *Py5cheSim* resulting throughput with the obtained from the theoretic throughput calculation tool from [42]. Simulation was made with one or more UEs with full buffer traffic and features support as shown in the next tables.

SU-MIMO

For SU-MIMO validation, one UE full buffer traffic simulation with different number of supported layers was executed, and obtained results were compared with those of the throughput calculation tool from [42], under the same conditions (Qm, Rmax, SCS, bandwidth, and useful symbols by slot). Tables 4.8 and 4.9 shows obtained results for a 10 MHz FDD cell, and 4.10 and 4.11 for a 100 MHz TDD cell.

Layers	Py5cheSim	Throughput Calculator	Relative error
4 layers	210 Mbps	222 Mbps	5%
8 layers	422 Mbps	446 Mbps	5%

Table 4.8: FDD DL Throughput comparison using SU-MIMO for 4 and 8 layers, and 10 MHz bandwidth.

When running the simulation with more UEs with the same SINR, in each case cell throughput remains as shown in comparison tables. Note that, at the moment the simulator assumes that all UEs supports the number of layers required, and the radio conditions are good enough. In real scenarios the use of SU-MIMO depends on UE capability, network configuration, and radio conditions.

Layers	Py5cheSim	Throughput Calculator	Relative error
4 layers	224 Mbps	238 Mbps	6%
8 layers	450 Mbps	476 Mbps	6%

Table 4.9: FDD UL Throughput comparison using SU-MIMO for 4 and 8 layers, and 10 MHz bandwidth.

Layers	Py5cheSim	Throughput Calculator	Relative error
4 layers	2033 Mbps	2154 Mbps	6%
8 layers	4026 Mbps	4310 Mbps	7%

Table 4.10: TDD DL Throughput comparison using SU-MIMO for 4 and 8 layers, and 100 MHz bandwidth.

Layers	Py5cheSim	Throughput Calculator	Relative error
4 layers	2226 Mbps	2366 Mbps	6%
8 layers	4429 Mbps	4730 Mbps	6%

Table 4.11: TDD UL Throughput comparison using SU-MIMO for 4 and 8 layers, and 100 MHz bandwidth.

MU-MIMO

The next tables shows comparison results for MU-MIMO tests. Comparison is made in terms of cell throughput for different configurations. First with 4 UEs with 4 MU-MIMO beams, next 2 UEs with 4 MU-MIMO beams, and finally, 4 UEs with 2 MU-MIMO beams.

Configuration	Py5cheSim	Throughput Calculator	Relative error
4 UEs/4 beams	212 Mbps	222 Mbps	5%
2 UEs/4 beams	211 Mbps	222 Mbps	5%
4 UEs/2 beams	106 Mbps	112 Mbps	5%

Table 4.12: FDD DL Throughput comparison using MU-MIMO for 2 and 4 UEs and beams, and 10 MHz bandwidth.

As can be seen, relative errors don't exceed the values shown in previous section. Note that when there are more MU-MIMO beams than users, the extra capacity is used to improve UE throughput, as expected. UEs SU-MIMO support is assumed. Also note that although MU-MIMO for TDD case is not considered, under the assumption of analogue beamforming use, the current implementation does not forbid it.

Layers	Py5cheSim	Throughput Calculator	Relative error
4 UEs/4 beams	233 Mbps	238 Mbps	2%
2 UEs/4 beams	226 Mbps	238 Mbps	5%
4 UEs/2 beams	113 Mbps	120 Mbps	6%

Table 4.13: FDD UL Throughput comparison using MU-MIMO for 2 and 4 UEs and beams, and 10 MHz bandwidth

Carrier Aggregation

To test CA feature implementation, again, *Py5cheSim* UE obtained throughput is compared with the obtained by Throughput Calculation Tool on [42]. Simulation is made for a one UE scenario with full buffer traffic, no MIMO use, and different CA configurations. Tables 4.14 , 4.15 , 4.16 and 4.17 presents the obtained results.

Note that, this implementation of CA has a big degree of simplification in the 3GPP procedures. First, as this is a cell level capacity simulator, the bandwidth increase is associated here to one cell. Furthermore, when configuring CA in the script simulation it is assumed that all UEs support the feature and the CC combination to simulate. The simulator makes no check of the configured carrier combination. In real scenarios different UEs could have different capabilities in terms of CA support, and band allocation could be made according to different policies configured in the node (cell load for example). Also, a finite list of possible combinations is available in each UE according to the 3GPP Release it supports. However, if all UEs support CA combination configured, have good coverage and there are no reasons to prioritize between the different supported carriers, as can be seen in the comparative tables, throughput results are similar to the theoretically expected. Having different cells for CA implementation improvement is considered for future work.

Configuration	Py5cheSim	Throughput Calculator	Relative error
10 + 10 MHz	106 Mbps	112 Mbps	5%
10 + 10 + 10 MHz	159 Mbps	166 Mbps	4%
20 + 20 MHz	216 Mbps	226 Mbps	4%

Table 4.14: FDD DL Throughput comparison using different CA combination examples.

Finally, simulations were repeated considering more than one UE, and cell throughput obtained is approximately equal to the obtained with one UE with full buffer traffic.

Configuration	Py5cheSim	Throughput Calculator	Relative error
10 + 10 MHz	115 Mbps	120 Mbps	4%
10 + 10 + 10 MHz	172 Mbps	178 Mbps	3%
20 + 20 MHz	232 Mbps	242 Mbps	4%

Table 4.15: FDD UL Throughput comparison using different CA combination examples.

Configuration	Py5cheSim	Throughput Calculator	Relative error
50 + 50 MHz	510 Mbps	538 Mbps	5%
50 + 50 + 50 MHz	763 Mbps	808 Mbps	6%
100 + 100 MHz	1019 Mbps	1078 Mbps	6%

Table 4.16: TDD DL Throughput comparison using different CA combination examples.

Configuration	Py5cheSim	Throughput Calculator	Relative error
50 + 50 MHz	568 Mbps	592 Mbps	4%
50 + 50 + 50 MHz	846 Mbps	886 Mbps	5%
100 + 100 MHz	1127 Mbps	1182 Mbps	5%

Table 4.17: TDD UL Throughput comparison using different CA combination examples.

4.2 Multi-Slice Validations

From here on, multi-slice validation is described. Note that, as at the moment there is no multi-slice simulator to compare with, validation will be made considering the expected results according to theory.

Multi-Slice validation is organized in two parts, according to the implementation structure. First, the slice creation and configuration according to the service requirements is validated. Then, Inter-slice schedulers operation is validated.

4.2.1 Slice Management Validation

Given the conceptual, simple and intuitive implementation of Network Slicing in this work, Slice management validation is made basically by checking the Slice configuration is according to the service requirements and the established mapping on *setInitialConfiguration* method from *Slice* Class. Note that different mapping between service requirements and Slice configuration can be implemented overwriting the later method. In this implementation Slice configuration is actually made in terms of:

- SCS: According to required delay
- AMC: Normal MCS allocation according to SINR or Robust

- Signalling Load: low or high
- Scheduling Algorithm: Set manually for each UE group in the simulation script
- DL/UL symbol allocation in TDD case
- Allocated PRBs: Resources are equally allocated to different Slices by default, but can change with other InterSlice scheduler implementations.

Except of resource allocation, Slice configuration is made at the creation moment, and remains unchanged during the simulation. It is assumed that some service requirements will not change during the simulation. However, as the number of UEs and traffic intensity could change, resource allocation between slices can be updated, with a configurable time granularity. Note that in real implementations there could be more parameters to configure within a Slice. However, it wasn't considered here, given the high degree of simplification in 3GPP procedures made in this implementation.

Table 4.18 shows the configured mapping between RAN Delay requirements and SCS configuration for a Slice.

Delay Requirement (ms)	≤ 2.5	≤ 5	≤ 10	> 10
SCS (kHz)	120	60	30	15

Table 4.18: Delay requirement to SCS mapping implemented in *dly2scs* method from *Slice* class.

As for signalling load, two levels were considered:

- Normal Signalling Load: used for eMBB and URLLC Slice types
- Low Signalling Load: used for mMTC Slice types

If the service requires high availability AMC algorithm is modified to allocate an MCS index lower than the expected for that SINR.

For Slice Management validation simulations were run using different service requirements, and Slice configuration is checked to be consistent with the criteria explained before:

As can be seen, Slices are configured according to service requirements and the established mapping in the *Slice* class. Resource Allocation between slices is equal in terms of bandwidth, but the number of PRB in each case respond to the numerology used. Note that even though 2ms delay is required for the URLLC-1 group, 60 kHz SCS is configured. This is because in this simulation the configured band is in FR1, and at least in R15, there is no support for 120 kHz SCS in FR1 bands. When running the same simulation in a TDD cell SCS for URLLC-1 UEs is 120 kHz, and 60 kHz for the other UE groups, as expected.

Slice Name	eMBB-1	mMTC-1	URLLC-1
Delay Requirement (ms)	10	20	2
Reliability Requirement			High
Allocated PRBs	17	35	8
SCS (kHz)	30	15	60
Signalling Load	Normal	Low	Normal
Robust MCS allocation	False	False	True

Table 4.19: Simulation with 3 UE groups in a 20 MHz cell. Service requirements vs Slice configuration.

4.2.2 Inter-Slice Scheduler Validation

Inter-Slice Scheduler validation is made by running simulations with more than one UEgroup, and checking that resource allocation between Slices is according to the scheduler algorithm used. Note that at the moment of writing this document, there are no multi-Slice simulators to compare with.

At the moment *Py5cheSim* presents three inter-slice schedulers:

- **Round Robin:** allocates the same amount of resources to the different slices, even if there is no traffic on UE bearer queues.
- **Round Robin Plus:** allocates the same amount of resources to the different slices, only with traffic in UE bearer queues. For example, if there are three Slices, but at the moment of scheduling resources between them, only two has UEs with traffic, band resources will be equally distributed between the other two.
- **Proportional Fair** with configurable exponents: allocates cell's resources to the Slice with the highest metric.

Note that if Slices have different numerology, even if the resources are distributed equally between them, PRBs available will be according to the configured numerology. However we can say that resources are equally distributed because Slices with higher SCS, will have proportionally shorter slots (in time duration).

Also note that no BWP support in UE group is checked in this implementation for inter slice resource allocation purposes. PRBs are assigned to a slice independently of the BWP support on the UEs. For example, if there is an UE group with low bandwidth support, scheduler can allocate more PRBs than the supported to this slice. Furthermore, this implementation does not check the frequency location of the allocated PRBs, and it is assumed that UEs supports the configured band in the simulation script, and can use all the PRBs assigned to the slice. This may be unrealistic for example with mMTC devices which will probably support lower bandwidths. It could be an improvement for next *Py5cheSim* versions. BWP support is only taken into account at intra slice scheduler level, to determine RBG size.

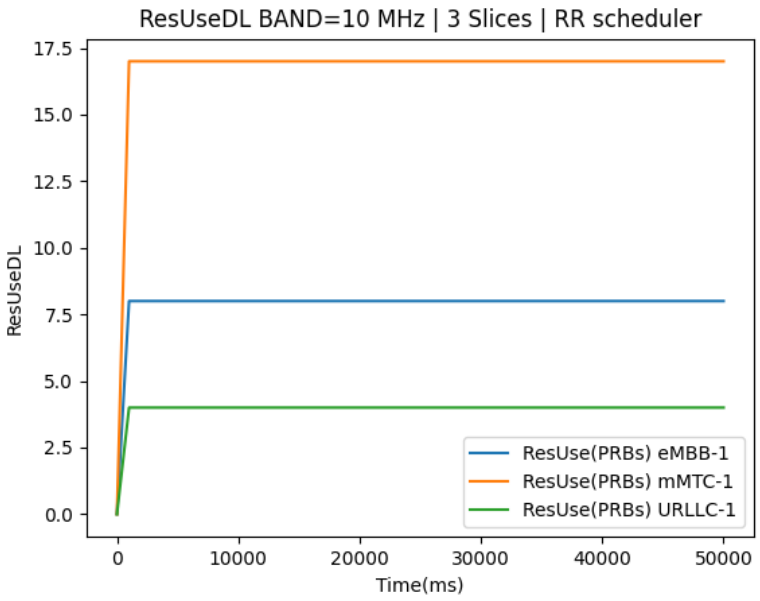

Figure 4.12: Resource allocation between slices using default Round Robin algorithm, in a 10 MHz FDD cell.

The following sections present different inter slice scheduling algorithm validation tests and its most meaningful results.

Round Robin scheduler

Round Robin inter slice scheduler was tested by running a simulation with 3 different UE groups with good SINR. UE groups have the same requirement shown in table 4.19, and Slice configuration is according to this.

Figure 4.12 shows resource allocation between the three slices. As can be seen, from the cell's 52 PRB (in the 15 kHz reference numerology) 17 are allocated to mMTC-1 slice (15 kHz SCS), 8 to eMBB-1 slice (30 kHz SCS), and 4 to URLLC-1 (60 kHz SCS). As the different slices has different numerologies, allocated PRB are different between them, but the resources are almost the same. The difference is explained by the trunking made by the inter slice scheduler implementation. The next thing to note is that PRB allocation doesn't change during the simulation, even when mMTC-1 slice has a very relaxed traffic profile.

Figure 4.13 shows total throughput for each slice. Note that only eMBB-1 slice is using the entire available slice bandwidth. In this case, obtained throughput is according to the expected for a 8 PRB allocation using 30 kHz SCS (17.3 Mbps from [42]). URLLC-1 and mMTC-1 traffic profiles have not enough intensity to consume their respective slices resources.

In case one of the UE groups has UEs with UL traffic, resource allocation remains unchanged. This is because in an FDD cell, RR scheduler allocates resources statically for DL and UL schedulers independently of there is traffic in one

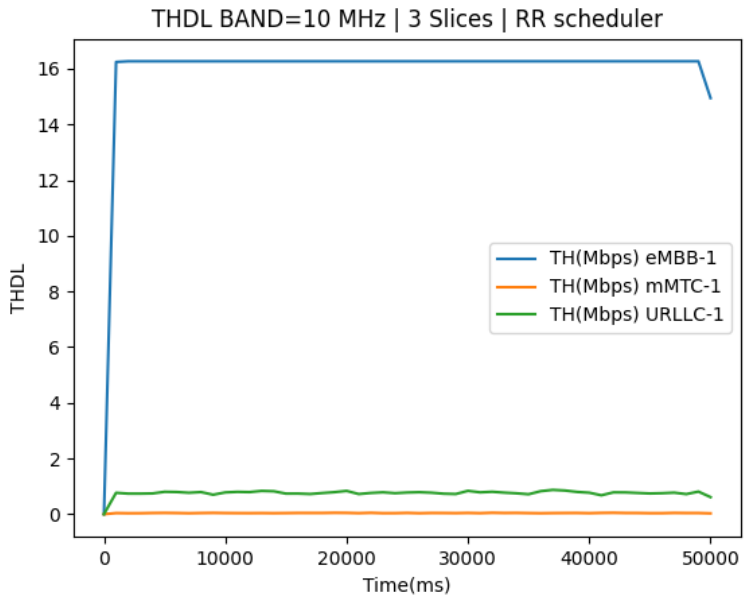

Figure 4.13: Slices throughput using default Round Robin algorithm, in a 10 MHz FDD cell.

direction or not.

When running the same simulation (same UEgroups) in a 50 MHz TDD cell, results are similar, as can be seen in figure 4.14 and 4.15. In this case, the band's 66 PRBs are assigned as follows: 22 for eMBB-1 slice with 60 kHz, 22 for mMTC-1 slice also with 60 kHz, and 11 for URLLC-1 slice with 120 kHz SCS, as expected. Again, resource allocation is static and independent of UEs traffic profile. Note that 60 kHz is used here as a base numerology because cell's band is in FR2, so 60 kHz SCS is valid for slices eMBB-1 and mMTC-1 delay requirements. However, as in FR2 120 kHz SCS is supported, URLLC-1 slice can use it to provide delays according to the configured requirement.

As for slice throughput, again, mMTC-1 and URLLC-1 slices are not expected to use the entire slices resources, so observed throughput is due to UEs traffic profile in this cases. In eMBB-1 slice case, as traffic profile is so much more intense, obtained throughput is the one 22 PRB with 60 kHz can give, in this case 85 Mbps (vs 90 Mbps according to [42]).

Again, in case one of the UE groups has UEs with UL traffic, resource allocation between slices remains unchanged. This is because in an TDD cell, RR scheduler also allocates resources statically for each slice, and the amount of symbols used for UL and DL in a slot is configured at slice level according to the traffic profile. Differences respect to the Figure 4.15 can be observed in UL/DL throughput if there are UL and DL users in the slice.

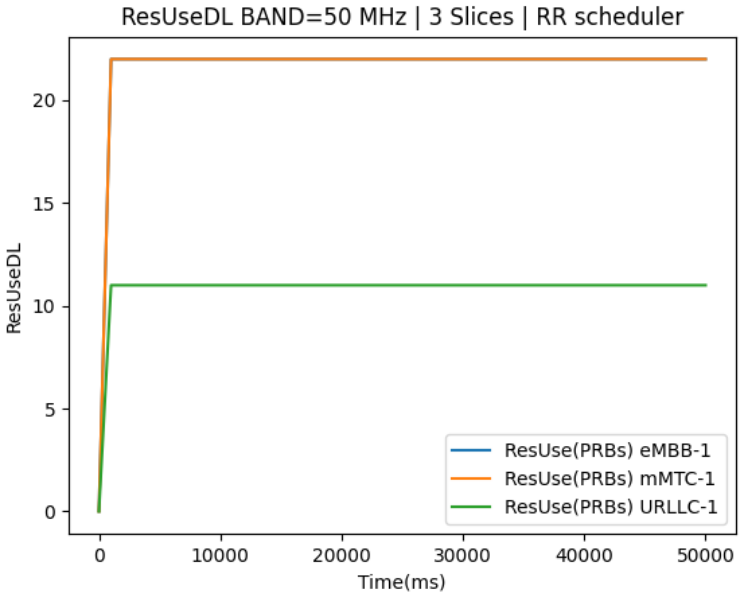

Figure 4.14: Resource allocation between slices using default Round Robin algorithm, in a 50 MHz TDD cell.

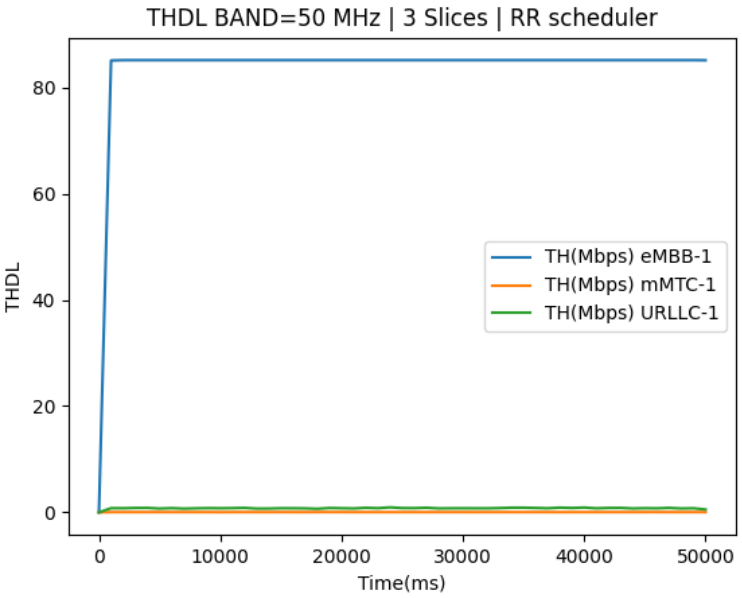

Figure 4.15: Slices throughput using default Round Robin algorithm, in a 50 MHz TDD cell.

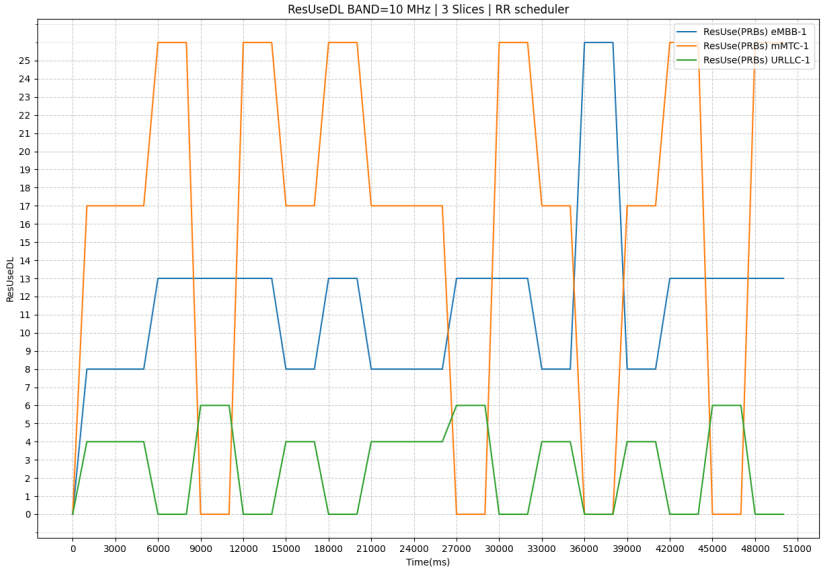

Figure 4.16: Resource allocation between slices using Round Robin Plus algorithm, in a 10 MHz FDD cell

Round Robin Plus Scheduler

For this scheduler validation, the later 3 Slices simulation is repeated with traffic profile and requirements from Table 4.19 and good radio conditions. Slices configuration remains the same, as expected because it does not depend of the InterSlice scheduler algorithm.

Figures 4.16 and 4.17 shows resource allocation between slices and slice's bearer buffer size in a 10 MHz FDD cell.

As can be seen in figures 4.16 and 4.17, although initially resource allocation is as with Round Robin algorithm, when there is no traffic from a slice, cell's PRBs are used for the remaining. As eMBB-1 slice has always traffic, it always gets resources from the cell. When mMTC-1 and URLLC-1 slices have not traffic (t=36000 ms), eMBB-1 slice takes all the cell's PRBs (26, using 30 kHz SCS). When there is not mMTC-1 slice traffic (at t=9000 ms for example), cell resources are shared between eMBB-1 and URLLC-1 slices equally, 13 PRB to eMBB-1 slice, and 6 to URLLC-1 (60 kHz SCS). When there is not URLLC-1 slice traffic (at t=12000 ms for example), cell resources are shared between eMBB-1 and mMTC-1 slices equally, again, 13 PRB to eMBB-1 slice, and 26 to mMTC-1 (15 kHz SCS). This is the expected behaviour of the implemented algorithm.

As can be seen in figure 4.18, when cell resources are assigned entirely to eMBB-1 slice (t=36000 ms), its obtained throughput is according to the expected for a 10 MHz cell (56 Mbps from [42]). When eMBB-1 slice has half of the cell resources, slice throughput decreases to approximately half of the former value (for

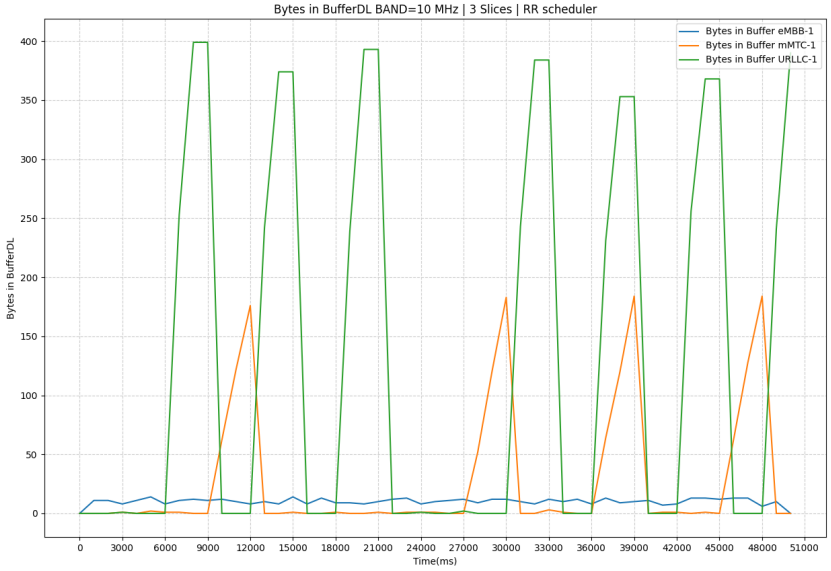

Figure 4.17: Bearer buffer in each slice using Round Robin Plus algorithm, in a 10 MHz FDD cell

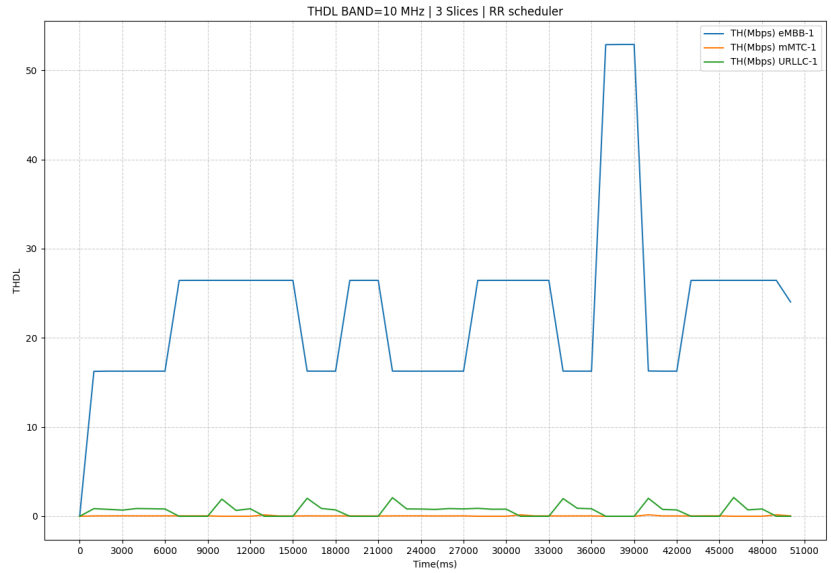

Figure 4.18: Slices Throughput (Mbps) using Round Robin Plus algorithm, in a 10 MHz FDD cell

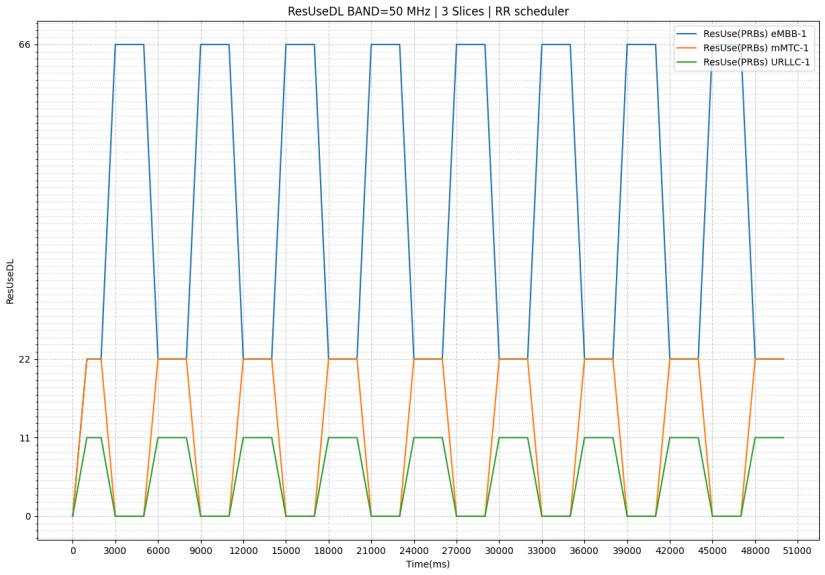

Figure 4.19: Resource allocation between slices using Round Robin Plus algorithm, in a 50 MHz TDD cell.

example at t=9000 ms). When it has a third part of the cell's resources, obtained throughput decreases, again proportionally. Note that throughput in the other slices respond to traffic profile and changes in resource allocation for the slice.

When running the same simulation (same UEgroups) in a 50 MHz TDD cell, results are similar, as can be seen in figures 4.19, 4.20, and 4.21. Almost the same comments than before apply here, with the following exception. As URLLC-1 and mMTC-1 slices traffic is very low in comparison with the allocated slices resources, the respective bearers buffers get empty quicker so there are more occasions in which eMBB-1 slice takes the entire band.

Proportional Fair

For this scheduler validation, the last 3 Slices simulation is repeated, with traffic profile and requirements from Table 4.19 and good radio conditions. Slices configuration remains the same, as expected because it does not depend of the InterSlice scheduler algorithm. As in IntraSlice case, three different exponent configuration were tested:

- Proportional Fair with numExp = 0 , and denExp = 1, PF01 from here on.
- Proportional Fair with numExp = 1 , and denExp = 1, PF11 from here on.
- Proportional Fair with numExp = 1 , and denExp = 0, PF10 from here on.

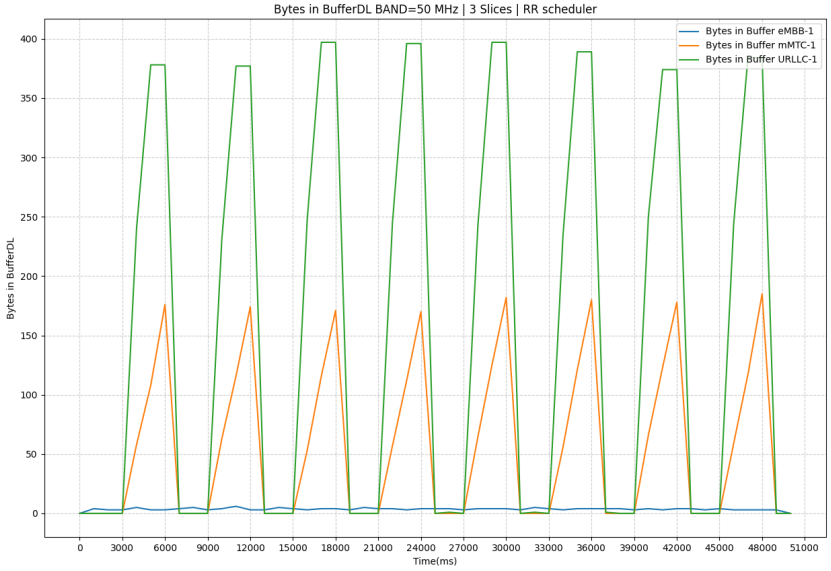

Figure 4.20: Bearer buffer in each slice using Round Robin Plus algorithm, in a 50 MHz TDD cell.

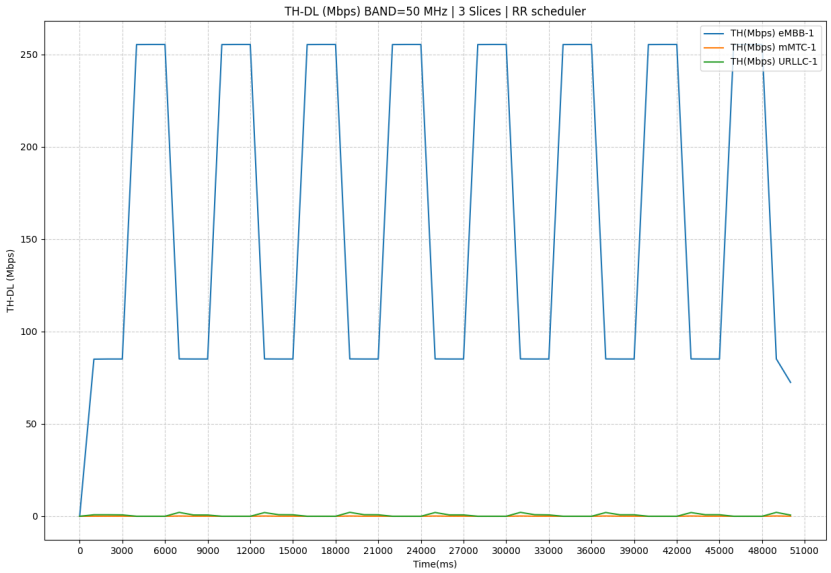

Figure 4.21: Slices Throughput (Mbps) using Round Robin Plus algorithm, in a 50 MHz TDD cell.

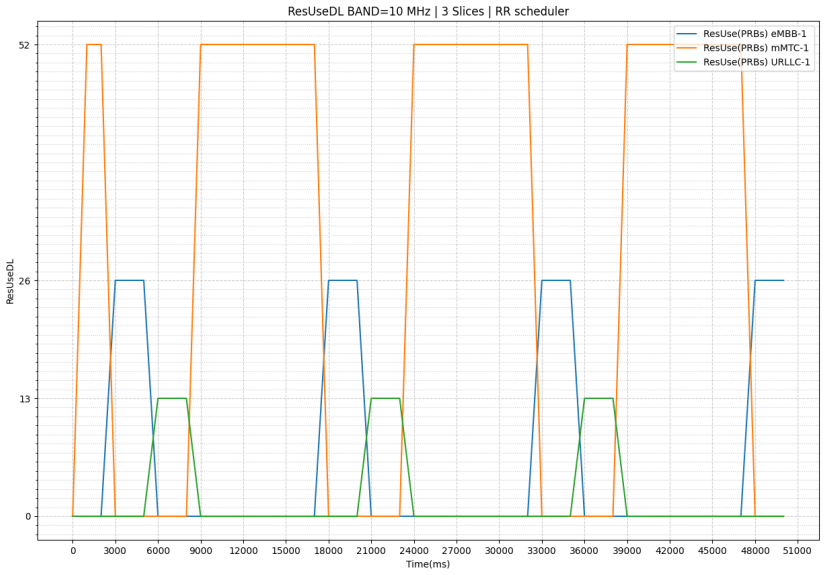

Figure 4.22: Resource allocation between slices using PF01 algorithm, in a 10 MHz FDD cell.

Figures 4.22 and 4.23 shows Resource allocation and metric for PF01 simulation. As can be seen, using this configuration mMTC-1 slice has the resources most of the time. This responds to the algorithm configuration and traffic profile. As mMTC-1 slice has the lightest traffic profile, received bytes easily gets lower than with the other slices. On the other hand, eMBB-1 and URLLC-1 slices occasionally gets resources, typically after mMTC-1 uses the resources for a while. This behaviour is expected considering that when configuring numExp = 0 , and denExp = 1 metric only depends on the slice past received bytes. Slices with high traffic will have lower metrics than slices with a light traffic profile, so will get resources less often.

Figures 4.24 and 4.25 shows resource allocation and metric for PF11 simulation. As can be seen, although mMTC-1 slice gets the resources more times than with PF01, it doesn't retain it longer. Again, this responds to the algorithm configuration and traffic profile. As mMTC-1 slice has the lightest traffic profile, received bytes easily gets lower than with the other slices, however when using numExp $>$ 0, other slices gets more chance of receive resources. In this case metric not only depends on slice received bytes. Here eMBB-1 and URLLC-1 slices have more chance to get resources. This behaviour is expected considering that when configuring numExp = 1 , and denExp = 1 metric depends on slice past received bytes and average UE TBS possible.

Figures 4.26 and 4.27 shows resource allocation and metric for PF10 simulation. Here, cell resources are taken most of the time by eMBB-1 and mMTC-1 slices. This responds to the algorithm configuration and slice conditions. As URLLC-1 has high availability requirements, allocated MCS is lower than in other slices, so

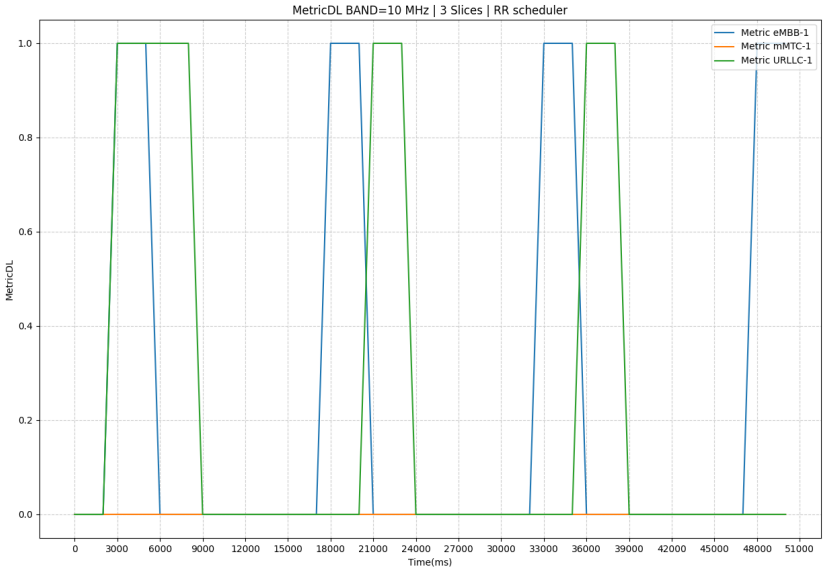

Figure 4.23: Metric results for each slice using PF01 algorithm, in a 10 MHz FDD cell.

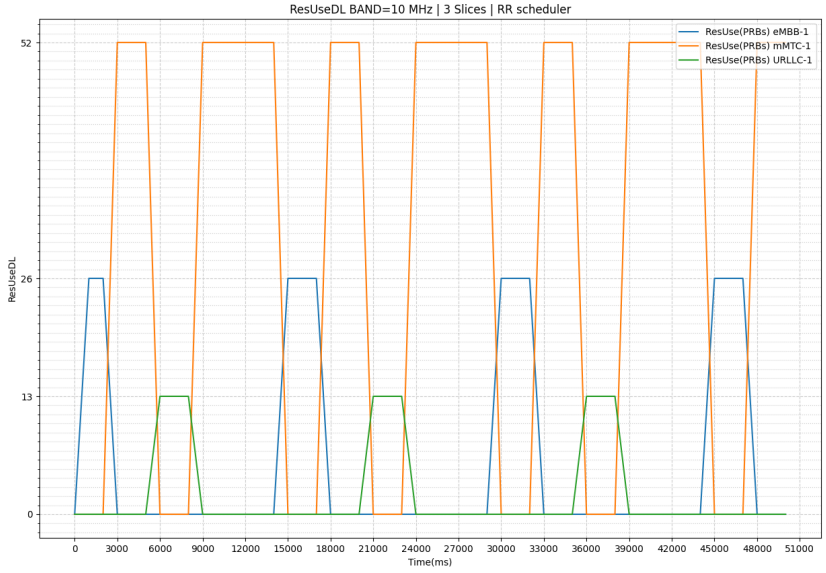

Figure 4.24: Resource allocation between slices using PF11 algorithm, in a 10 MHz FDD cell.

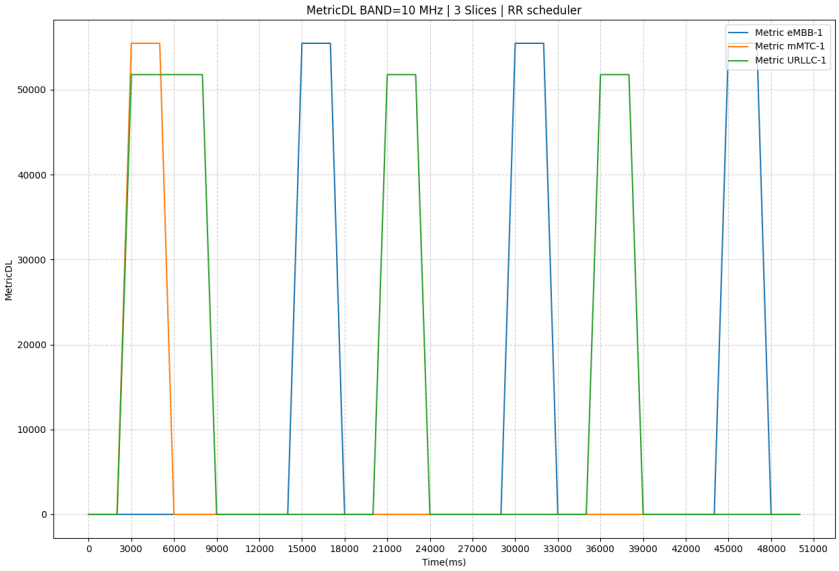

Figure 4.25: Metric results for each slice using PF11 algorithm, in a 10 MHz FDD cell.

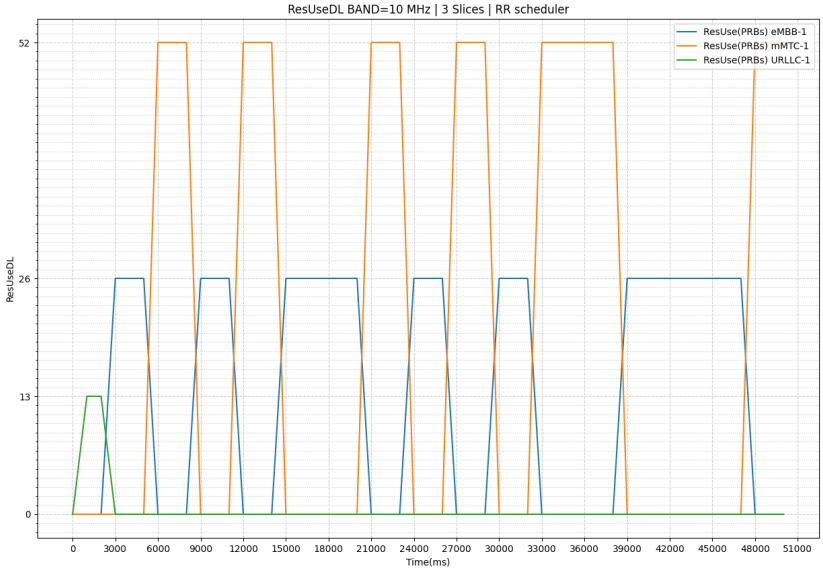

Figure 4.26: Resource allocation between slices using PF10 algorithm, in a 10 MHz FDD cell.

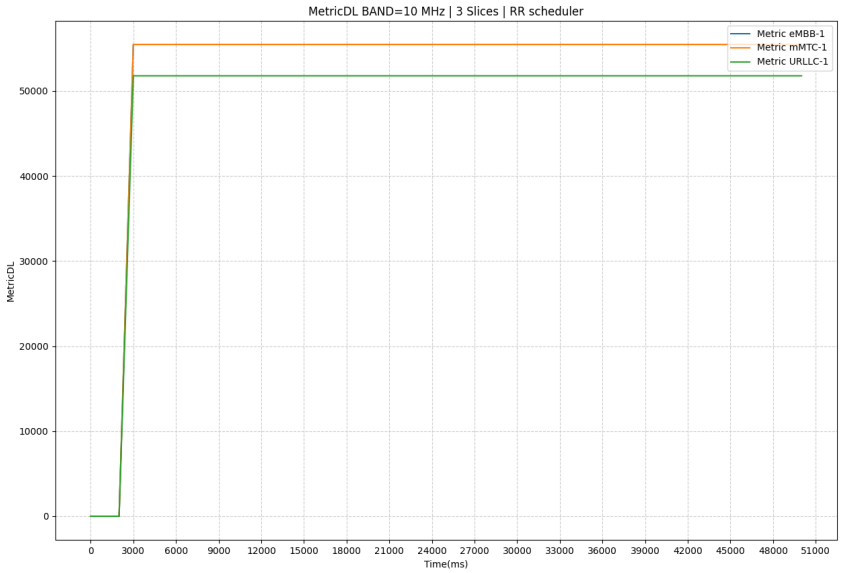

Figure 4.27: Metric results for each slice using PF10 algorithm, in a 10 MHz FDD cell.

average UE TBS will be lower. As metric in this case only depends on posisble average UE TBS, URLLC-1 slice will always have lower metric, so it won't get resources. The expeption to this rule if when the simulation begins, because all slices metric is initialized in 0, so a random slice is selected to gets the cell resources. As for the other slices, cell resources are allocated between them randomly, because there are no differences in availability requirements nor radio conditions among them, so they will have the same metric.

Almost the same results are obtained when running the same simulation in TDD, with the exception of the difference in the available bandwidth.

4.3 Validation Conclusion

The obtained validation results shows that even with the high level of simplification made on this implementation, differences with those obtained from *5G-LENA* and theoretic values are under the considered error margin. Furthermore, different scheduler algorithms were tested at intra and inter slice levels showing results according to the expected. In this way, the developed tool is validated and considered adequate for different scheduling algorithms analysis, as long as the defined error margin could be tolerated. More accurate results could be reached enriching the defined models, adding, however, more complexity to the implementation.

4.4 Chapter Summary

In this chapter *Py5cheSim* implementation was validated at different levels. At Slice level, validation tests were made using different available tools in terms of MCS allocation, TBS calculation and resulting UE throughput for different SINR levels. Also different scheduling algorithms were tested at intra and inter Slice levels, showing results according to the expected, and validating *Py5cheSim* as a "skeleton" tool for different 5G scheduling algorithms analysis, from a cell capacity point of view. New schedulers can be easily integrated by adding a subclass of the basic schedulers (at intra or inter slice level) and overwriting the *resAlloc* method, following the PF scheduler implementation example. The next chapter shows some examples of use cases followed by the thesis conclusion.

Chapter 5

Py5cheSim Use Cases Examples

In this chapter a few simple use case examples are presented to show the utility of the developed tool. First, PF scheduler at intra slice level is tested on traffic profiles different than MBB/eMBB. Then, the different inter slice schedulers implemented are tested in an scenario with different slices with different traffic profiles and requirements.

5.1 PF Scheduler Evaluation for Traffic Profiles other than eMBB

In this example, PF scheduler is evaluated for traffic profiles different from the traditional MBB/eMBB. There has been several works showing the benefits of using PF schedulers for MBB traffic profiles. Given the new services envisioned by 5G, is interesting to have a first evaluation of the well known schedulers, already studied for the traditional traffic profiles.

5.1.1 IoT Traffic Profile Case

In this example simulation with one UE group is configured with the next characteristics:

- 100 UEs with UL traffic
- Packet size: 350 bytes
- Packet inter-arrival time: 60 sec
- SINR between 2 and 8 dB
- One slice with only one PRB available, with the objective to easily stress the cell.

This traffic profile is similar to the typically found in IoT applications (small packets sporadically sent to a server). A more detailed traffic profile can be built

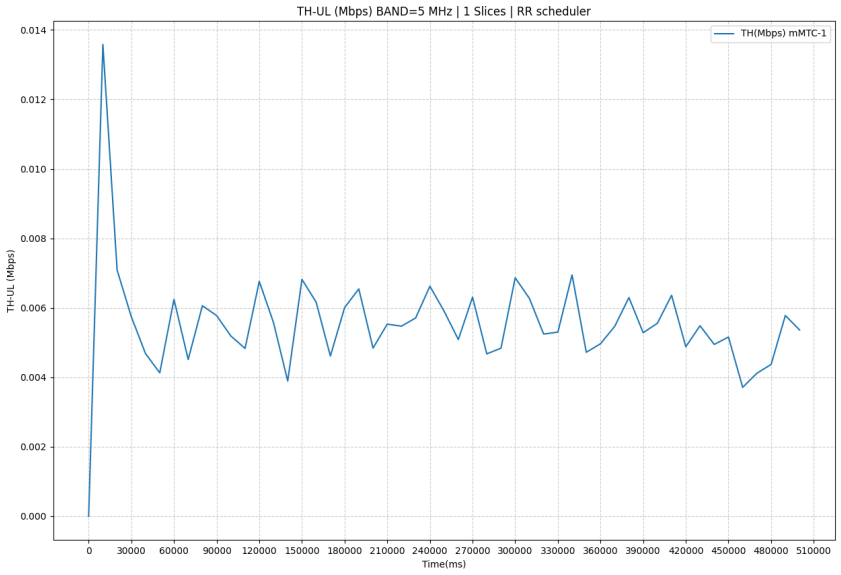

Figure 5.1: Cell throughput for 100 IoT devices simulation using RR scheduler.

modifying *getPsize* and *getParrRate* on *PacketFlow* class. Note that low SINR are configured to use low MCS, again trying to get closer to an IoT scenario.

At high level, scheduling for IoT services should be agile enough to avoid overload the buffers in both IoT devices and network. In IoT devices because typically this kind of terminals are cheap and simple, and probably doesn't have too much storage capacity. At network level it is important due to the massive amount of this devices is intended to support. A good measure of agility in the scheduling can be done by measuring the bearer buffer size.

Figures 5.1, 5.2 and 5.3 show simulation results using RR scheduler. The simulation results show that cell capacity has not been exceeded, at least from a UP perspective. However there are packets in buffer during most of the simulation. Note that Random Access procedures and PHY Control Channel capacity evaluation is not possible with *Py5cheSim* at the moment, so cell capacity evaluation here is made in terms of UP data transport capacity.

Figures 5.4, 5.5 and 5.6 show simulation results using PF11 scheduler. In this case again, cell capacity has not been exceeded as can be seen in the figures. However the amount of packets in buffer is lower than with RR scheduler. Resources are not exactly equally distributed between UEs as can be seen in Figure 5.5, according to the expected from PF11 scheduler. Although cell throughput has almost no impact in this case, PF11 scheduler has improved the scheduling agility resulting better than RR in this case for massive connection low intensity traffic profiles. Note that for IoT traffic scheduling other considerations could be made from another design aspects, as for example, device power consumption. Given the high degree of simplification made in this implementation, this kind of consid-

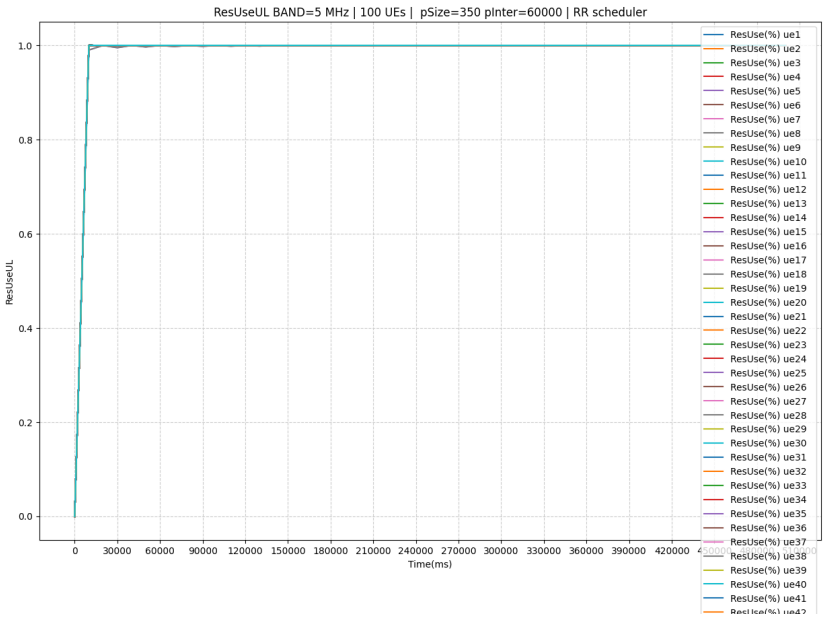

Figure 5.2: Resource Allocation for 100 IoT devices simulation using RR scheduler.

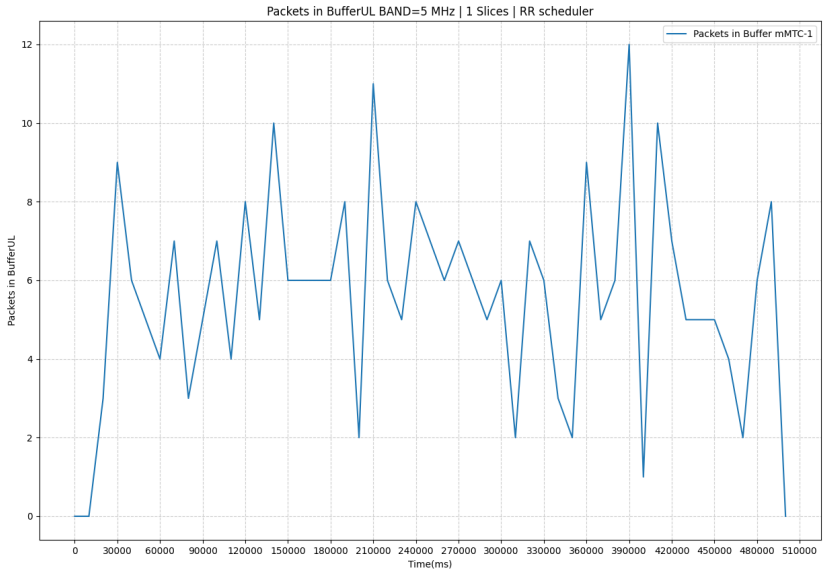

Figure 5.3: Buffer size for 100 IoT devices simulation using RR scheduler.

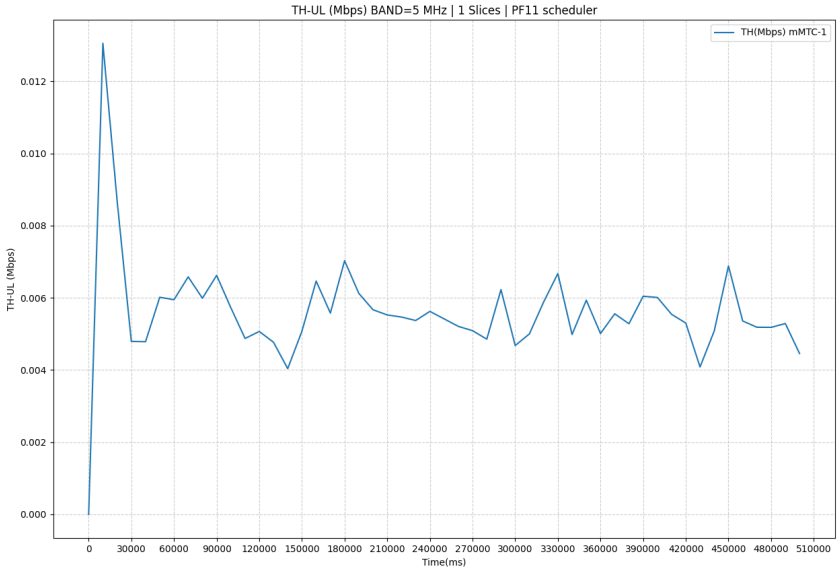

Figure 5.4: Cell throughput for 100 IoT devices simulation using PF11 scheduler.

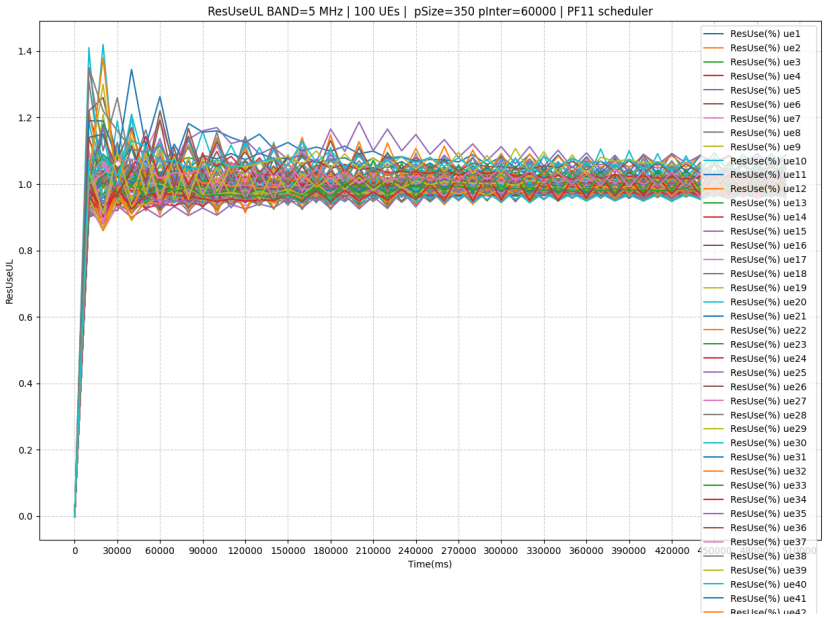

Figure 5.5: Resource Allocation for 100 IoT devices simulation using PF11 scheduler.

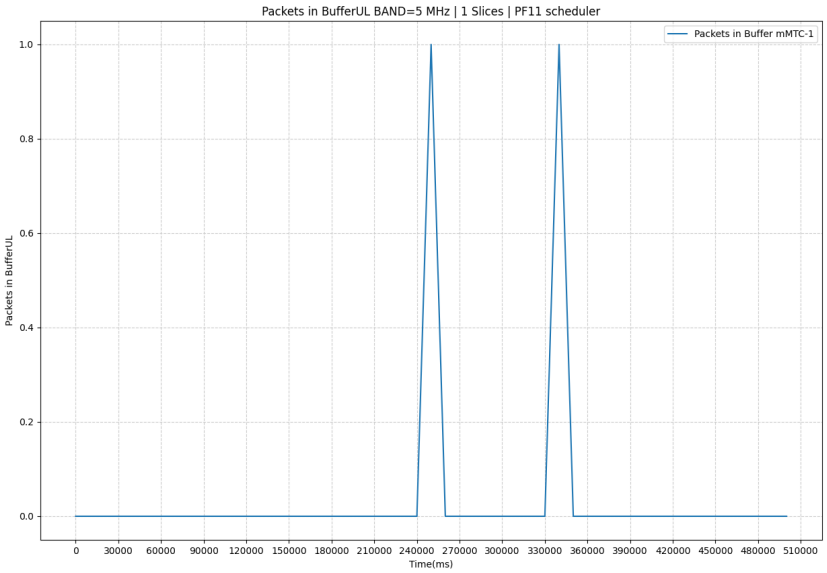

Figure 5.6: Buffer size for 100 IoT devices simulation using PF11 scheduler.

erations are out of the scope of this work. Again, a more detailed analysis could be made adding more complexity to the developed tool. However, note that the obtained results are valid for a primary analysis, prior to define a research line to follow, for example.

5.1.2 URLLC Traffic Profile Case

In this example simulation with one UE group is configured with the next characteristics:

- 6 UEs with UL traffic
- Packet size: 1500 bytes
- Packet inter-arrival time: 6 ms
- SINR between 10 and 30 dB
- One slice in a 10 MHz cell.

This traffic profile is similar to the typically found in video vigilance applications (average quality video streaming traffic profile).

Scheduling for this kind of applications should be agile enough to provide a low delay. As *Py5cheSim* actually does not take delay measures, agility in scheduling will be measured again trough bearer buffer size.

Figures 5.7, 5.8 and 5.9 show simulation results using RR scheduler. As can

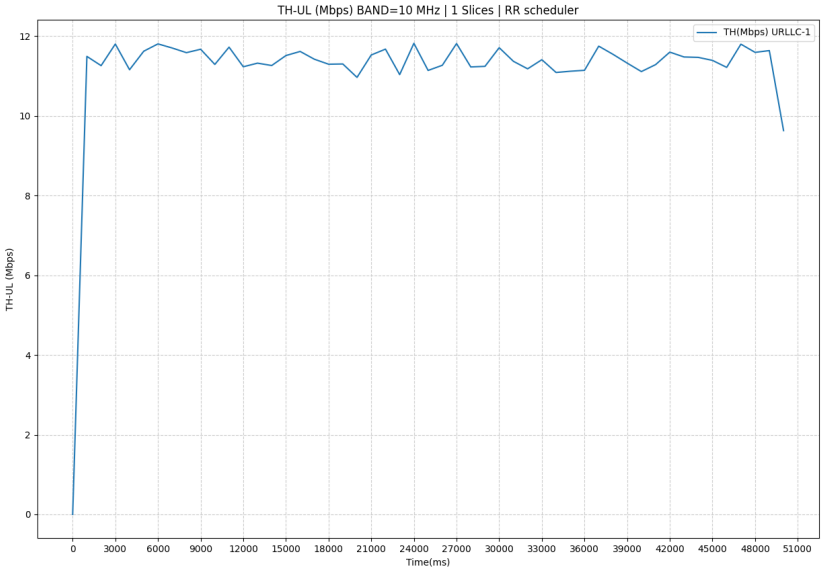

Figure 5.7: Cell throughput for 6 video cameras simulation using RR scheduler.

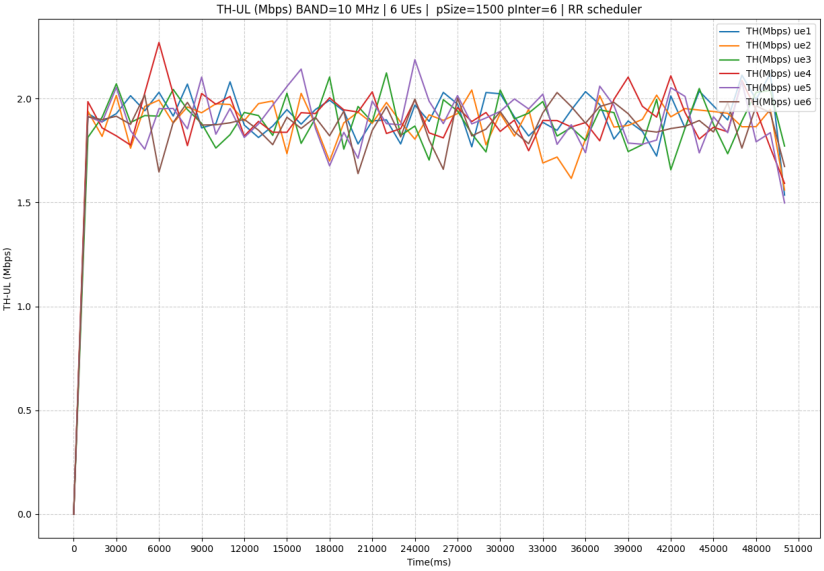

Figure 5.8: UE throughput for 6 video cameras simulation using RR scheduler.

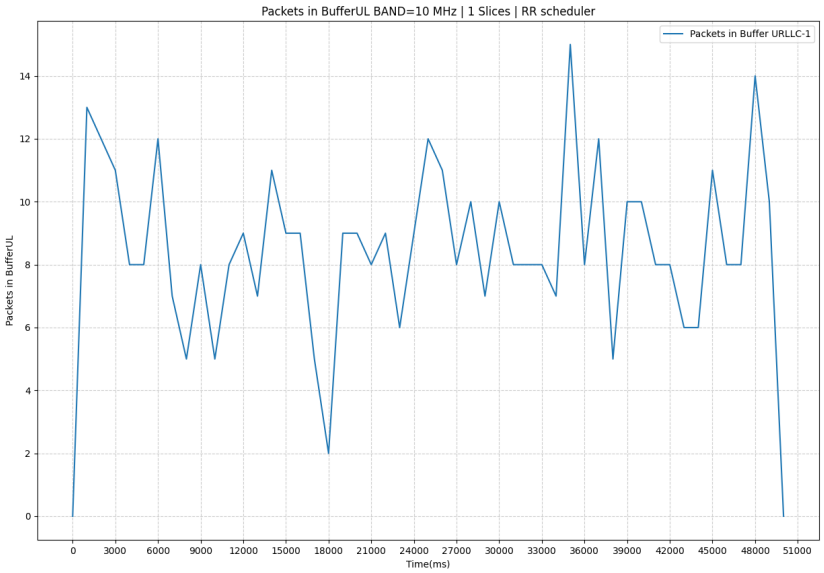

Figure 5.9: Buffer size for 6 video cameras using RR scheduler.

be seen in Figure 5.9, as bearer buffer size is contained during the simulation, cell capacity was not exceeded from an UP perspective. When running the same simulation using PF11 scheduler, there is no improvement, as can be seen in Figures 5.10, 5.11 and 5.12. However, if the same simulation is repeated using 60 kHz SCS, bearer buffer size is reduced, as can be seen in figure 5.13. This behabiour is expected considering that using a bigger numerology reduces TTI, so scheduling can be done more frequently, which improves delay and makes packets stay less longer on buffers. Cell and UE throughput has no meaningful changes. If we repeat the same simulation again using 60 kHz SCS and PF11 scheduler bearer buffer size is improved again, as can be seen in Figure 5.14. Again, cell and UE throughput has no meaningful changes.

This example shows that even though cell and UE throughput have almost no impact, the use of a higher numerology and PF11 scheduler improves the scheduling agility for the configured traffic profile. Again although a more detailed analysis could be done, the obtained results are an interesting input for a future research.

5.1.3 Example Conclusion

In this section implemented PF scheduler has been tested on two scenarios different than the typical MBB. The obtained results show that PF scheduling could be beneficial for the considered examples based on a primary analysis made through one of the measurements reported by *Py5cheSim*. More complexity can be added to the development to enrich the analysis. Furthermore, other schedulers can be easily implemented and integrated to the developed tool to consider more scheduler

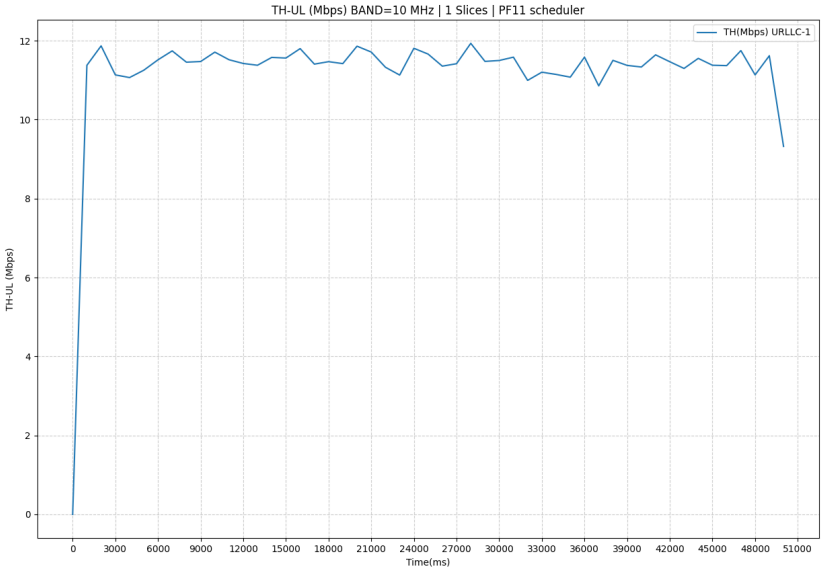

Figure 5.10: Cell throughput for 6 video cameras simulation using PF11 scheduler.

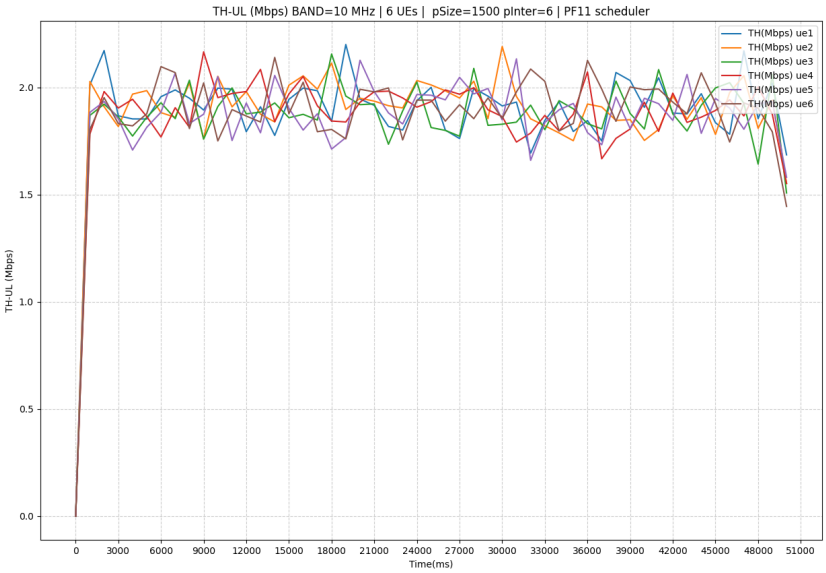

Figure 5.11: UE throughput for 6 video cameras simulation using PF11 scheduler.

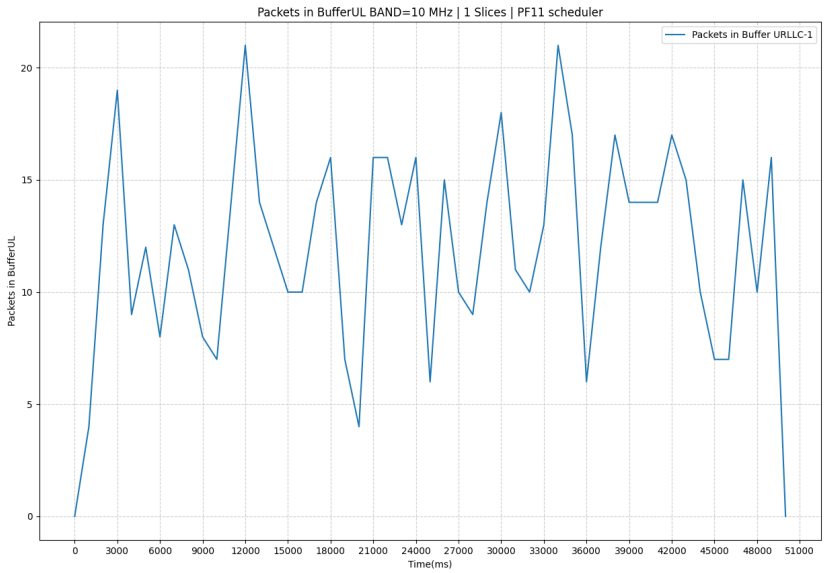

Figure 5.12: Buffer size for 6 video cameras using PF11 scheduler.

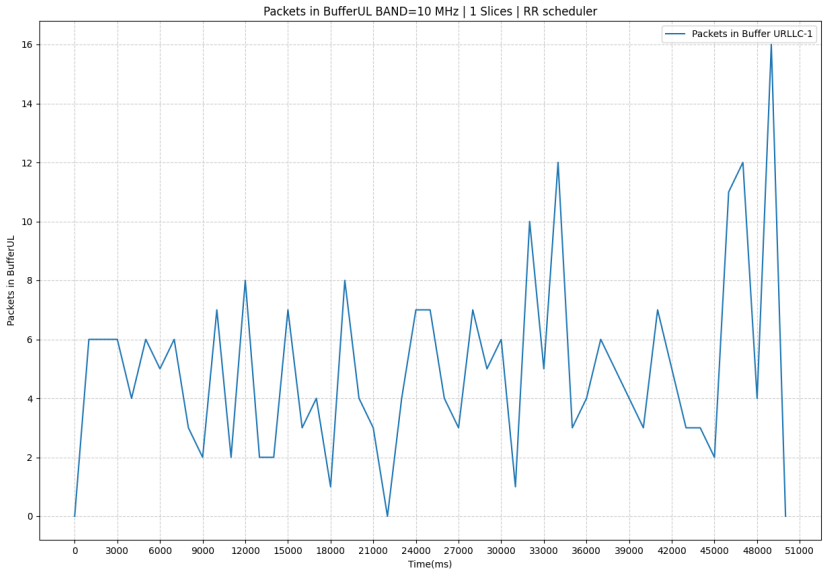

Figure 5.13: Buffer size for 6 video cameras using RR scheduler and 60kHz SCS.

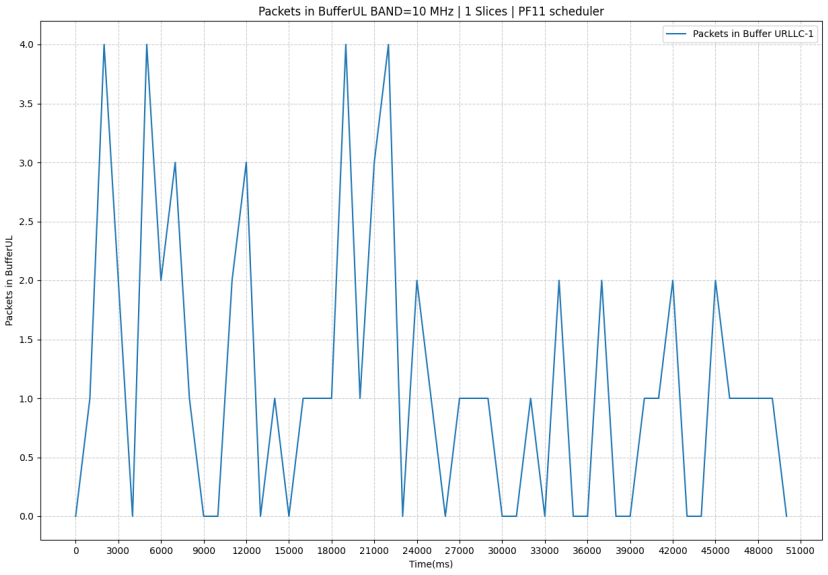

Figure 5.14: Buffer size for 6 video cameras using PF11 scheduler and 60kHz SCS.

options.

5.2 Multiplexing Different Services in one Cell

In this example different implemented inter slice schedulers are evaluated for multiplexing different services on the same cell. A *Py5cheSim* simulation is configured with four UE groups as can be seen in Table 5.1. All UE groups are configured to

Slice Name	eMBB-1	eMBB-2	mMTC-1	URLLC-1
Delay Requirement (ms)	20	20	20	5
UE number	4	15	150	8
Average SINR (dB)	25	20	5	25
Packet Size (bytes)	2000000	5000	350	1500
Packet Arrival Rate (ms)	5000	10	6000	6
SCS (kHz)	15	15	15	60

Table 5.1: 4 UE groups/slices configuration for test simulation within a 20 MHz FDD cell.

send UL traffic. Slice eMBB-1 will be handling bursty traffic trying to simulate video or high quality image uploads. Slice eMBB-2 will have similar smartphone background traffic with relative small packets sent every few ms. m-MTC-1 slice will be handling a big number of devices sending small and infrequent packets to

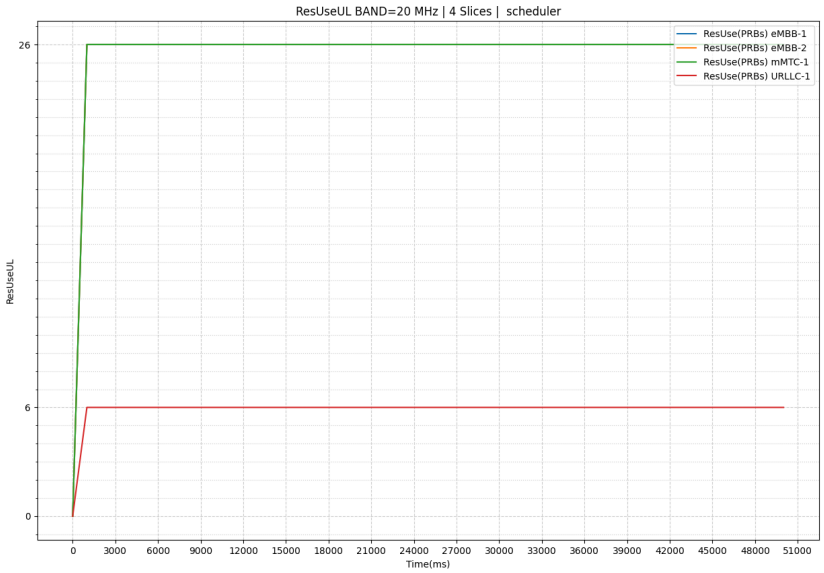

Figure 5.15: Four slices resource allocation in a 20 MHz cell using RR inter slice scheduler.

a server (similar to the first example of this chapter). Finally, URLLC-1 slice will be handling eight UEs with video vigilance camera similar traffic profile (the same tested in the later example).

As for the scheduling requirements, slices mMTC-1 and URLLC-1 will need agile packet scheduling given the devices and service requirements. Although eMBB slices will be benefited by agile scheduling this is not a requirement.

5.2.1 Round Robin Scheduler Simulation

Figures 5.15, 5.16 and 5.17 show resource allocation, throughput and bearer buffer size for each slice using RR scheduler for inter slice scheduling, and PF11 as intra slice scheduler. Resource allocation between Slices is static and even as can be seen in Figure 5.15, according to the configured inter slice scheduler. Slice throughput responds to the different traffic profiles configured, and cell's UP capacity was not exceeded, as can be seen in 5.17. However all slices shows packets in bearer buffer at some point during the simulation. Also, there is no so much room to support more devices in the mMTC-1 slice if one doesn't want to increase the amount of buffered packets.

5.2.2 Round Robin Plus Scheduler Simulation

Figures 5.18, 5.19 and 5.20 show resource allocation, throughput and bearer buffer size for each slice using RRplus scheduler for inter slice scheduling. Resource allocation between Slices in this case takes into account slice bearers buffer status

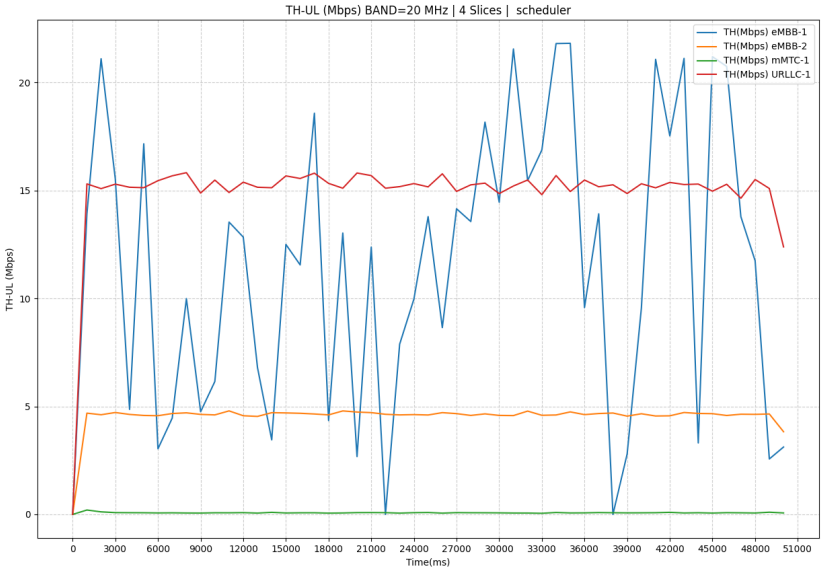

Figure 5.16: Four slices throughput in a 20 MHz cell using RR inter slice scheduler.

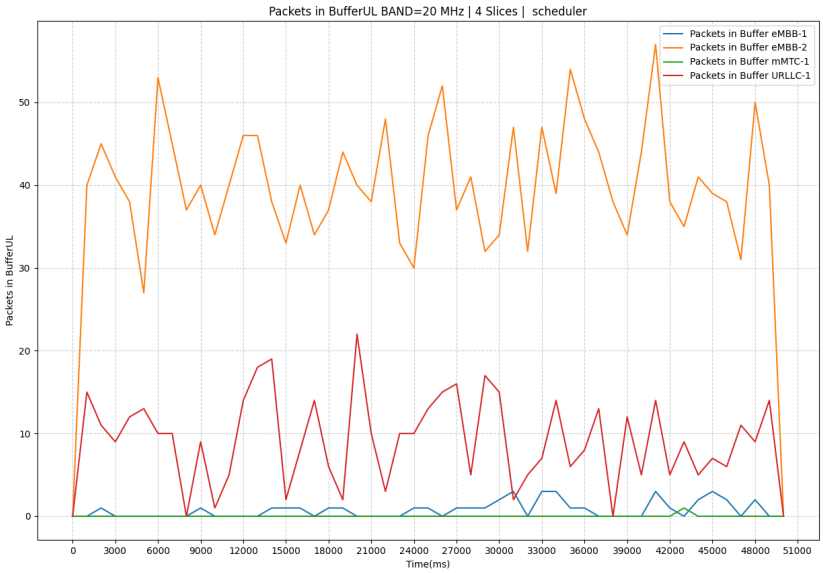

Figure 5.17: Four slices buffer size in a 20 MHz cell using RR inter slice scheduler.

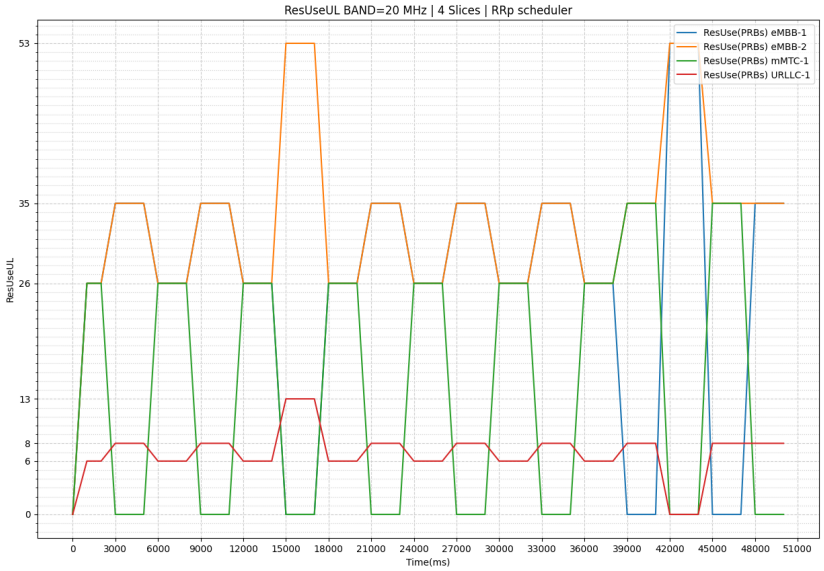

Figure 5.18: Four slices resource allocation in a 20 MHz cell using RR Plus inter slice scheduler.

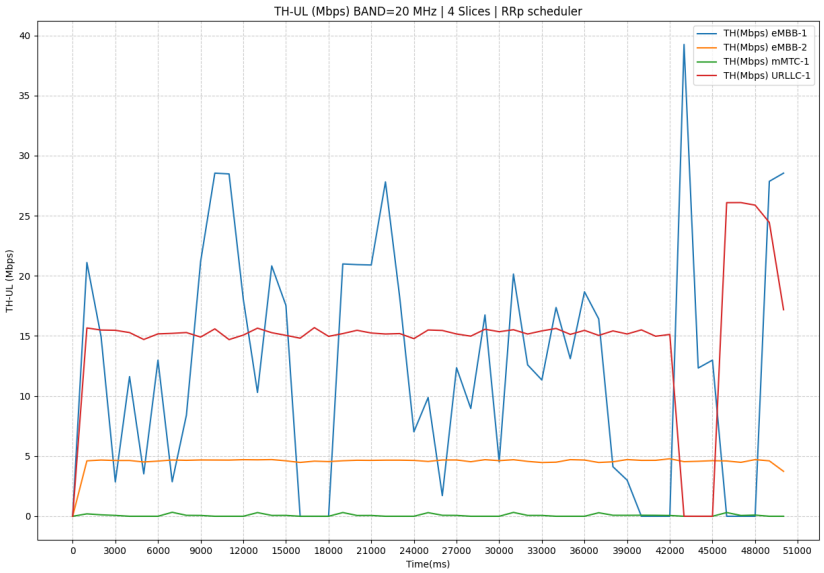

Figure 5.19: Four slices throughput in a 20 MHz cell using RR Plus inter slice scheduler.

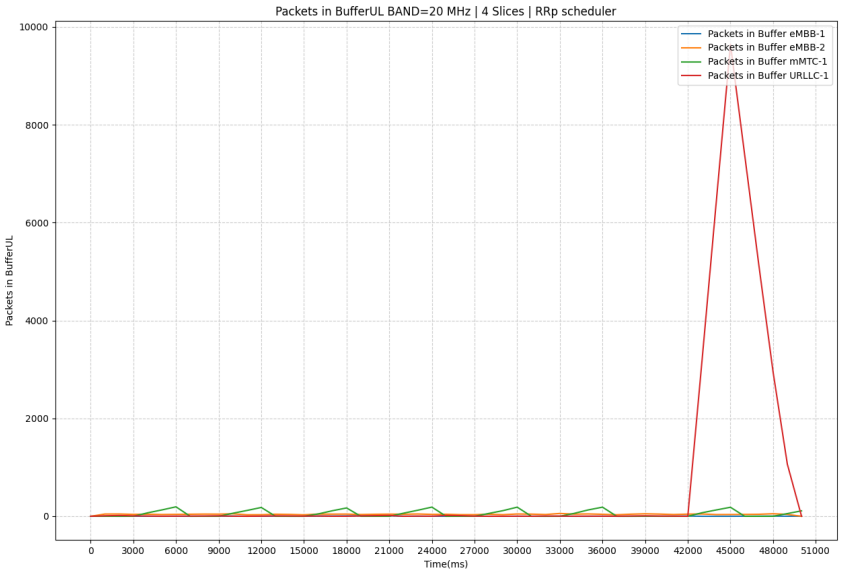

Figure 5.20: Four slices buffer size in a 20 MHz cell using RR Plus inter slice scheduler.

as can be seen in Figures 5.18 and 5.20. Slice throughput responds to the different traffic profiles configured, and cell's UP capacity was not exceeded, as can be seen in 5.20. Furthermore, eMBB-1 slice reach higher throughput than with RR scheduler when there are no packets in buffer for other slices, improving user experience. However when there are not packets in buffer for the mMTC or URLLC-1 slices, as RR Plus inter slice scheduler does not allocate PRBs for this slices, bearer buffer increases quickly as can be seen in Figures 5.20 and 5.21, given the configured traffic profile. Although eMBB-1 slice shows an improvement in throughput, as mMTC-1 and URLLC-1 slices requires agile scheduling, RR Plus scheduler could not be the best choice when having this kind of services. Another thing interesting to note is that eMBB-2 slice, as is handling packets more frequently always has resources even though is the slice with more relaxed scheduling constraints. RR plus scheduler, as gives resources based on buffer status in some way is prioritizing this kind of traffic profile, which is no desirable.

5.2.3 PF11 Scheduler Simulation

Figures 5.22, 5.23 and 5.24 show resource allocation, throughput and bearer buffer size for each slice using PF11 scheduler for inter slice scheduling.

In this case, as cell resources are entirely allocated to the slice with the highest metric, there are some intervals in which eMBB-2 and URLLC-1 slices has no resources, so packets will be buffered, as can be seen in the figures 5.22 and 5.20. However Figure 5.23 shows an important increase in eMBB-1 slice throughput. Again, given the scheduling requirements for URLLC-1 slice, this implementation

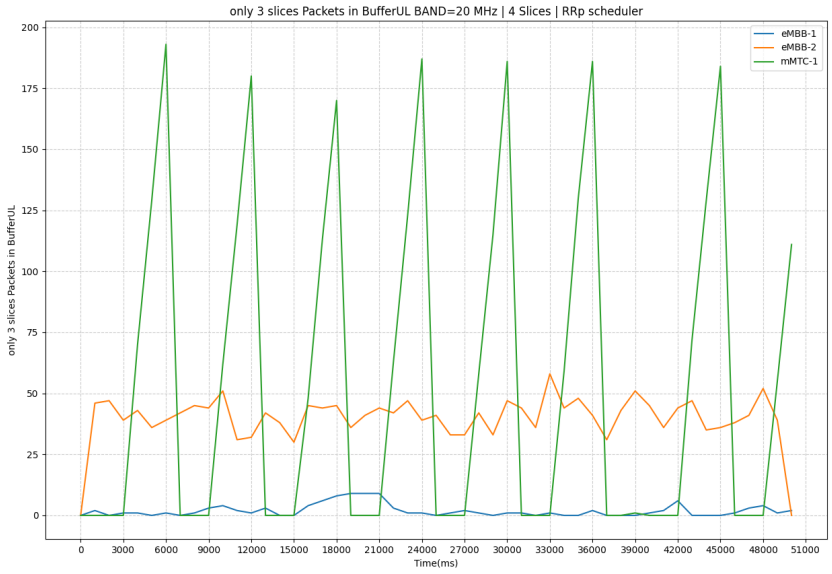

Figure 5.21: eMBB-1, eMBB-2 and mMTC-1 slices buffer size in a 20 MHz cell using RR Plus inter slice scheduler.

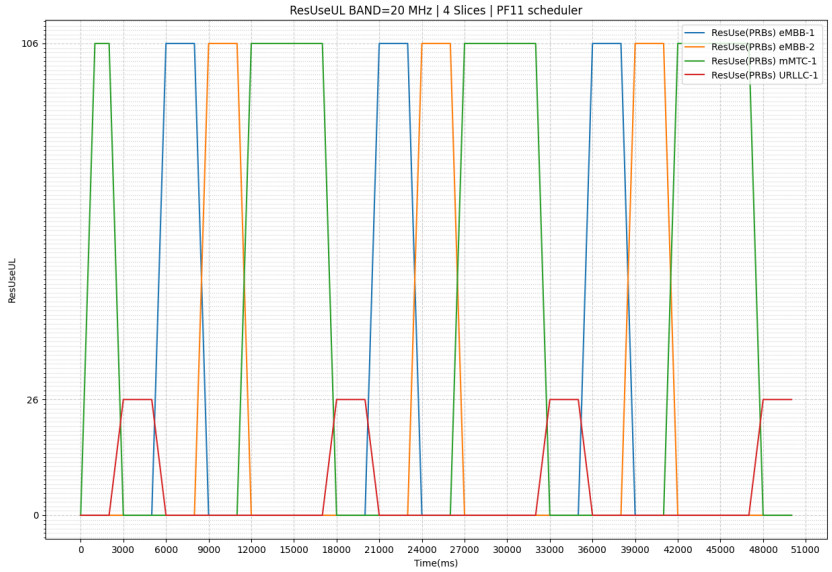

Figure 5.22: Four slices resource allocation in a 20 MHz cell using PF11 inter slice scheduler.

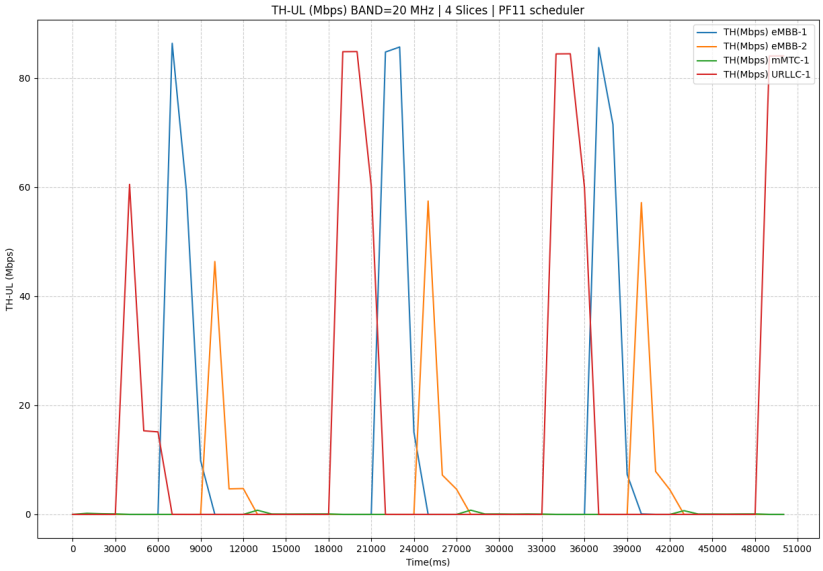

Figure 5.23: Four slices throughput in a 20 MHz cell using PF11 inter slice scheduler.

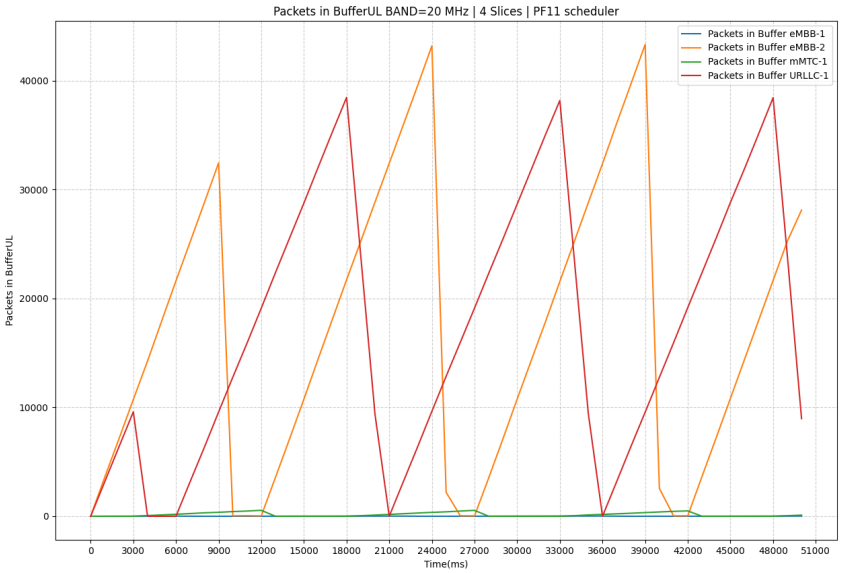

Figure 5.24: Four slices buffer size in a 20 MHz cell using PF11 inter slice scheduler.

of PF11 scheduler doesn't provide better results than the Round Robin scheduler.

As a general observation for this use case is interesting to note that the simplest solution when there is always resources for services that require agile scheduling results better than others which applies more intelligence. RR scheduler simply divides the band equally between the configured slices without considering the traffic profile or service requirements. The other implemented schedulers fail for applications with agile scheduling requirements because they could not give resources to a slice for a while if suddenly does not detect packets to transmit in buffers. A possible solution could be to reduce the inter slice scheduler time granularity. Another possibility is to consider a dynamic resource allocation scheme when PRBs are divided between slices according to the buffer size. Also Proportional Fair metric could be adapted to consider the buffer size. However, given the delay constraints of expected URLLC services, possibly the best solution for this kind of services comes with punctured mini-slot scheduling to reduce as much as possible the time to wait in buffer for each packet.

5.2.4 Example Conclusion

In this example three different inter slice scheduling options were tested in a scenario with 4 slices with different traffic profiles and scheduling requirements. Although Proportional Fair resulted better for intra slice test examples, for inter slice scheduling of services with delay or buffer capacity constraints it does not result better than Round Robin. The same occurs with the implemented Round Robin Plus scheduler, which considers buffers size for scheduling decision. It is important to note that although the developed inter slice schedulers did not work as good as expected, *Py5cheSim* allows the development of new inter slice schedulers easily to continue with the analysis.

This page has been intentionally left in blank

Chapter 6

Thesis Conclusion

In this work a light, free and simple Python tool for 5G scheduling analysis was built. The main concepts behind the technology were introduced, and the developed tool features, architecture and design considerations were presented. Also the main results of validation tests were shown, followed by a few simple use cases.

The obtained validation test results, along with the use case examples shows the potential of *Py5cheSim* for different schedulers performance primary analysis. New different scheduling algorithms can be easily implemented and evaluated along with the already built ones. Furthermore, given the availability of machine learning tools for Python, new AI based scheduling algorithms can be easily integrated. The information given by the tool even though quite basic, is useful to understand the scheduling operative and reach a primary conclusion. Although there is a lot to improve in terms of the considered models, this first *Py5cheSim* version reaches the objective to be a scheduler analysis base tool which serves as a starting point for future 5G scheduling research.

As for the traditional scheduling algorithms analysis for the new 5G scenarios, although more evaluation tests should be done, the obtained primary results show that for intra slice scheduling Proportional Fair could improve Round Robin scheduler when considering the new traffic profiles. However for inter slice scheduling the primary analysis does not show the same. Other considerations should be taken for this kind of scheduling given the different requirements of the services to multiplex. First, buffer size should be considered when defining a metric for resource allocation to favor agile scheduling. Besides, inter slice scheduling granularity should be studied in terms of implementation possibilities, and performance improvement vs processing load. Also AI based algorithms could offer a better solution, considering the characteristic of the mobile traffic as shown in [43].

Given the high delay and availability constraints of URLLC services it is important to note that mini-slot punctured scheduling could provide better results than traditional inter slice scheduling. Although the first version of *Py5cheSim* does not support punctured mini-slot scheduling, given the actual TDD scheduler implementation, mini-slot scheduling could be easily integrated. Future versions could offer this kind of scheduling after a few changes in the *TDD_scheduler* class. Another interesting thing to note is that even though in the example use case IoT

traffic was seen as one more slice, actually there are no native NR IoT devices. 3GPP actual devices are still Cat-M or NB-IoT ones. 3GPP R16 provides a way of coexistence between this types of devices with NR traffic based on the standardized for LTE. However simulation tools should be open and prepared to support more things than the actually available by the standard. Future *Py5cheSim* versions could integrate LTE-M and NB-IoT support easily by adding new MCS tables, adapting TBS, and isolating the required PRBs in each case.

Also, as mentioned before, more improvements could be made in terms of the implemented models. Better channel and error models could be added to provide more realistic results. MCS allocation could also be improved to consider the later. Interaction between all this concepts could be better modeled taking into account the dynamic nature of mobile traffic. More than one bearer by UE could be supported to include ACK packets for example. Also scheduling modeling could be improved to consider the standard timing parameters to add delay measures for a better evaluation of scheduling algorithms in URLLC traffic cases. Carrier Aggregation and MIMO implementation could also be improved to consider more realistic test cases. As for Slice management, mapping between service requirements and configuration can be improved considering traffic load changes during the simulation. Finally, lower layers signalling could be considered for capacity evaluation behind the overheads included in TBS calculation. All this improvements could make the simulation results more accurate, but could also add more complexity to the developed tool, and possibly require more processing capacity and/or time to run a simulation. Furthermore, the lightweight of this version along with its easiness to use and to extend are some of the main advantages that *Py5cheSim* presents in comparison with *5G-LENA*. *Py5cheSim* can be easily installed and used in a regular PC with Ubuntu. A simulation can be easily configured adding a few lines in the simulation script, in Python. The simulation can take from seconds to minutes depending on the number of UEs and their configured traffic profile. KPI charts are automatically generated after running the simulation and stored along with simulation statistics. Although the results obtained with *5G-LENA* can be more accurate, running a simulation requires to build a C++ simulation script following the *ns3* and *nr* objects structure, so it is recommended to have knowledge and experience with the tool before using it. A simulation could take from seconds to hours in a regular PC depending on the number of UEs and their configured traffic profile. Besides, adding new schedulers is out of the possibilities of someone with no *ns3* developing experience or at least C++ knowledge and a lot of time to learn about *ns3* and its involved modules.

As for this thesis evaluation in terms of the experience itself a few things can be added. At the beginning of this work there was not an appropriate simulation tool to study 5G scheduling even at a basic level. The standard was in developing phase. In that context, the idea of creating a new simulator even maybe too ambitious, was considered reasonable. Note that the complete version of *5G-LENA* was released six months ago, while the first was available in middle 2020. The level of detail found in *Py5cheSim* models responds to the dedicated resources to the project. From that resources, there was a part mostly at the project beginning which was

almost entirely dedicated to learn about the technology itself. Then a first LTE prototype was built and tested against the *lte* module of the *ns3* simulator, which was the only available tool for validation at that moment. Learning to use the *lte* module took much more time than the expected, given the complexity of the tool. As validation results were good enough to continue, the prototype was adapted to consider 5G introduced features and differences in radio interface processing, having been already released the first version of 3GPP Release 15 specifications. Finally, the last *Py5cheSim* version was validated during the first part of this year using mainly the *nr* module released six months ago. The amount of time resources dedicated to validation, and particularly to learn how to use *ns3* modules for this project was really meaningful. The simulation tool is strongly oriented to C++ developers. That had impact on available resources for model improvement, project documentation and research itself.

This page has been intentionally left in blank

Appendix A

Py5cheSim User Manual

In this annex a brief guide to *Py5cheSim* use is presented.

A.1 Previous Steps

Before use the developed tool a few things need to be done:

- Check the installed python version. *Py5cheSim* runs in Ubuntu using Python 3.
- Install *SimPy* version 3 or 4. The easiest way is using python pip. In ubuntu this could be done with the followin commands:

  sudo apt-get install python3-pip
  pip3 install simpy

- Install matplotlib and python-tk packages. Again in ubuntu this could be done by running:

  python3 -mpip install matplotlib
  apt-get install python-tk

- Unzip and copy the simulator folder in any place on /home/your-user-name.

A.2 Configuring a Simulation

Although *Py5cheSim* actually has nine modules, only the *simulation.py* module should be edited to run a simulation. The simulator allows to configure different things, from cell parameters to traffic profiles for the simulation, as can be seen next.

A.2.1 Cell Parameters

This are parameters relative to cell configuration as for example:

- Frequency Range (FR1 or FR2)
- Cell bandwidth. It is configured as a list with as much element as CC will be used in the simulation.
- TDD operation. True if the cell is TDD false if it is FDD.
- Buffer size (bytes). This parameter is used to define when a packet is lost.
- Inter Slice Scheduler algorithm. Round Robin will be used by default.

A.2.2 Simulation parameters

This parameters are relative to the simulation configuration as for example:

- Simulation duration (ms).
- Debugging mode option. When set to true, debugging files are generated in html format with detailed scheduling operative for each Slice and UE.
- Measurement interval (ms). The time granularity for the statistics reports.
- Inter slice time granularity (ms).

A.2.3 UE Traffic Profiles

The simulation runs for a defined set of UE groups, each one with a defined traffic profile. Also each UE group will be mapped to a Slice, which will be configured according to the requirements set on the UE group. The UE group list should contain all UE groups the simulation is configurad for.

Each UE group must be configured with:

- Number of users (DL or UL)
- Traffic profile in terms of packet size (bytes) and inter-arrival rate (ms), according to the model specified in chapter 3. For DL simulation only DL traffic profile parameters must be set. The same occurs with UL traffic.
- Slice label, for slice identification purposes.
- Delay requirement in ms. This would be the required delay limit at RAN level, and will be used to set slice SCS configuration.
- Availability requirement, can be high or normal. By default is normal.
- Intra slice scheduler algorithm: RR for Round Robin, and PFXY for Proportional Fair with numExp=X and denExp=Y.
- MIMO mode. Can be SU or MU depending on the type of MIMO scheme to use.

```
gaby@gaby-desktop:~/Documentos/Mestria/Tesis/Simulador$ python3 simulation.py
[=         ] 10% complete simulation
[==        ] 20% complete simulation
[===       ] 30% complete simulation
[====      ] 40% complete simulation
[=====     ] 50% complete simulation
[======    ] 60% complete simulation
[=======   ] 70% complete simulation
[========  ] 80% complete simulation

-------------------------------------------------
                SLICE: eMBB-1
-------------------------------------------------

Accumulated UL indicators by user:
ue1      Sent Packets:25 Lost Packets:0
        SINRav: 25 MCSav: 21 PLR: 0.0 % Throughput: 2.67
ue2      Sent Packets:30 Lost Packets:0
        SINRav: 25 MCSav: 21 PLR: 0.0 % Throughput: 3.43
ue3      Sent Packets:28 Lost Packets:0
        SINRav: 25 MCSav: 21 PLR: 0.0 % Throughput: 4.21
ue4      Sent Packets:34 Lost Packets:0
        SINRav: 24 MCSav: 20 PLR: 0.0 % Throughput: 3.52
Average UL Indicators:
Packet Loss Rate av:    0.0 %
Throughput av:  3.46 Mbps
Connections av:         4
Slice Resources:        106 PRBs
Symbols in slot:        14
Slice Numerology:       15 kHz
Configured Signalling Load: 1e-06
Using Robust MCS: False
```

Figure A.1: Simulation output example.

- Number of layers in case of MIMO use.
- UEs SINR to consider during the simulation codified as a string. The first character must be S in case all UEs has the same SINR, or D if different UE should use different initial SINR values. Next goes the value of the UEs initial SINR in the first case, or the maximum value of initial SINR, in the second. For example, in a simulation with all UEs with the same initial SINR equal to 28 dB will be coded as S28.

A.3 Running a Simulation

To run a simulation, after configuration on the *simulation.py* module, from a linux terminal in the directory in which *Py5cheSim* was saved, simply run:

python3 simulation.py

Additionally, *Py5cheSim* will generate:

- A Figures folder in the current directory with different kpi measured charts.
- Intra and inter slice statistics files with the collected raw data.
- Intra and inter slice html files with the event logging for debugging purposes.

This page has been intentionally left in blank

Bibliography

[1] Sigcomm n2women workshop 2021. *https://conferences.sigcomm.org/sigcomm/2021/n2women.html*, 2021.

[2] XLVII confererencia latinoamericana en informatica clei 2021. *https://clei2021.cr/home*, 2021.

[3] NGMN alliance. NGMN 5G withe paper.

[4] Hiroyuki Atarashi Anass Benjebbour, Koshiro Kitao. Imt-2020 radio interface standardization trends in itu-r. *NTT DOCOMO Technical Journal Vol.19 No.3 (Jan, 2018)*, 2018.

[5] J. Baek, J. Bae, Y. Kim, J. Lim, E. Park, J. Lee, G. Lee, S. I. Han, C. Chu, and Y. Han. 5g k-simulator of flexible, open, modular (fom) structure and web-based 5g k-simplatform. In *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–4, 2019.

[6] Matlab Help Center. ofdmmod - ofdm modulation.

[7] Christopher Cox. An introduction to lte. 2014.

[8] Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). NR Module, release 1.1. 2021.

[9] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). The lena ns-3 lte module, release v8. 2014.

[10] C. Rattaro G. Pereyra and P. Belzarena. Py5chesim. `https://github.com/ClaudinaRattaro/Py5cheSim`, 2021.

[11] Martín Rodriguez Gustavo Bounous, Leticia Silva. Modelado y planificación de redes LTE - LTEst. 2012.

[12] http://www.sharetechnote.com/. RF - SNR vs SINAD. *http://www.sharetechnote.com/html/RF_Handbook_SNR.html*.

[13] Michael Hudson-Doyle and various contributors. Pydoctor documentation. *https://pydoctor.readthedocs.io/en/latest/index.html*, 2020.

[14] ITU-R. Recommendation ITU-R M.2083-0: IMT Vision – framework and overall objectives of the future development of IMT for 2020 and beyond. 2015.

[15] Stefan Parkvall Erik Dahlman Asbjørn Grøvlen-Christian Hoymann Dirk Gerstenberger Janne Peisa, Patrik Persson. 5G evolution: 3GPP Releases 16 and 17 overview. *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-nr-evolution*, 2020.

[16] For 3GPP Jeanette Wannstrom. Carrier aggregation explained. *https://www.3gpp.org/technologies/keywords-acronyms/101-carrier-aggregation-explained*, 2013.

[17] F. Kaltenberger, G. d. Souza, R. Knopp, and H. Wang. The openairinterface 5g new radio implementation: Current status and roadmap. In *WSA 2019; 23rd International ITG Workshop on Smart Antennas*, pages 1–5, 2019.

[18] Florian Kaltenberger, Aloizio P. Silva, Abhimanyu Gosain, Luhan Wang, and Tien-Thinh Nguyen. Openairinterface: Democratizing innovation in the 5g era. *Computer Networks*, 176:107284, 2020.

[19] Taehyoung Kim Karol Schober Kazuki Takeda, Huilin Xu and Xingqin Lin. Understanding the heart of the 5G air interface: An overview of physical downlink control channel for 5G new radio (NR).

[20] Y. Kim, J. Bae, J. Lim, E. Park, J. Baek, S. I. Han, C. Chu, and Y. Han. 5g k-simulator: 5g system simulator for performance evaluation. In *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–2, 2018.

[21] Adlen Ksentini and Navid Nikaein. Toward enforcing network slicing on ran: Flexibility and resources abstraction. *IEEE Communications Magazine*, 2017.

[22] Peter J Smith Thomas Haustein-Peiying Zhu Prasan De Silva Fredrik Tufvesson Anass Benjebbour Gerhard Wunder Mansoor Shafi, Andreas F. Molisch. 5g: A tutorial overview of standards, trials, challenges, deployment and practice. *IEEE Journal on Selected Areas in Communications*, 2017.

[23] Michele Polese Russell Ford Sourjya Dutta-Sundeep Rangan Michele Zorz Marco Mezzavilla, Menglei Zhang. End-to-end simulation of 5g mmwave networks. *IEEE Communication Surveys and Tutorials*, 2018.

[24] Martin Klaus Müller, Fjolla Ademaj, Thomas Dittrich, Agnes Fastenbauer, Blanca Ramos Elbal, Armand Nabavi, Lukas Nagel, Stefan Schwarz, and Markus Rupp. Flexible multi-node simulation of cellular mobile communications: the Vienna 5G System Level Simulator. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):17, September 2018.

[25] Giovanni Nardini, Dario Sabella, Giovanni Stea, Purvi Thakkar, and Antonio Virdis. Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks. *IEEE Access*, 8:181176–181191, 2020.

[26] Giovanni Nardini, Giovanni Stea, Antonio Virdis, and Dario Sabella. Simu5g: A system-level simulator for 5g networks. 07 2020.

[27] Lorenza Giupponi Biljana Bojovic Natale Patriciello, Sandra Lagen. The impact of nr scheduling timings on end-to-end delay for uplink traffic. *IEEE Global Communications Conference 2019*.

[28] 3GPP Technical Specification Group Radio Access Network. TS 36.213 evolved universal terrestrial radio access (E-UTRA); physical layer procedures, Release 15.

[29] 3GPP Technical Specification Group Radio Access Network. TS 38.214 NR physical layer procedures for data, Release 15.

[30] 3GPP Technical Specification Group Radio Access Network. TS 38.300 nr and ng-ran overall description; stage 2, Release 15.

[31] 3GPP Technical Specification Group Radio Access Network. TS 38.306 NR user equipment (ue) radio access capabilities, Release 15.

[32] Natale Patriciello, Sandra Lagen, Biljana Bojovic, and Lorenza Giupponi. An e2e simulator for 5g nr networks. *Simulation Modelling Practice and Theory*, 96:101933, 2019.

[33] Stefan Pratschner, Bashar Tahir, Ljiljana Marijanovic, Mariam Mussbah, Kiril Kirev, Ronald Nissel, Stefan Schwarz, and Markus Rupp. Versatile mobile communications simulation: the Vienna 5G Link Level Simulator. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):226, September 2018.

[34] Jordi Pérez-Romero Ramon Ferrús, Oriol Sallent and Ramón Agustí. On 5g radio access network slicing: Radio interface protocol features and configuration. *IEEE Communications Magazine*, 2018.

[35] Zwi Altman Ana Galindo-Serrano Salah Eddine Elayoubi, Sana Ben Jemaa. 5g ran slicing for verticals: Enablers and challenges. *IEEE Communications Magazine, Institute of Electrical and Electronics Engineers*, 2019.

[36] Atanu Halder Naidu Mullaguru Sam Guirguis, Kausik Ray Chaudhuri. M2M communications: Enablement in 4g lte, deployment considerations and evolution path to 5g. 2017.

[37] 3GPP Technical Specification Group Services and System Aspects. TR 21.915 release 15 description; summary of rel-15 work items, Release 15.

[38] 3GPP Technical Specification Group Services and System Aspects. TR 21.916 release 16 description; summary of rel-16 work items, Release 16.

[39] 3GPP Technical Specification Group Services and System Aspects. TR 23.799 study on architecture for next generation system, Release 14.

[40] Team SimPy. Documentation for simpy. *https://simpy.readthedocs.io/en/latest/contents.html*, 2020.

[41] Javier Campos. Keysight Technologies. Understanding the 5G NR physical layer.

[42] From 5G-tools.com Vinogradov Oleg. 5G NR Throughput calculator. *https://5g-tools.com/5g-nr-throughput-calculator/*.

[43] Daizo Ikeda Ken Ohta Tadanori Mizuno Yu Abiko, Takato Saito and Hiroshi Mineno. Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning. *IEEE Access*, 2020.

[44] S. M. A. Zaidi, M. Manalastas, H. Farooq, and A. Imran. Syntheticnet: A 3gpp compliant simulator for ai enabled 5g and beyond. *IEEE Access*, 8:82938–82950, 2020.

List of Tables

List of Tables

List of Figures

This page has been intentionally left in blank

Abbreviations

3GPP	3rd Generation Partnership Project
5GC	5G Core
AMC	Adaptive Modulation and Coding
AMF	Authentication and Mobility Function
AR	Augmented Reality
BER	Bit Error Rate
BLER	Block Error Rate
BWP	Bandwidth Part
CA	Carrier Aggregation
CC	Component Carrier
CORESET	Control Resource Set
CP	Control Plane
CQI	Channel Quality Indication
CSI-RS	Channel State Information Reference Signal
D/A	Digital to Analogue
DC	Dual Connectivity
DL	Downlink
DMRS	Demodulation Reference Signal
DRB	Data Radio Bearer
DSS	Dynamic Spectrum Sharing
eMBB	Enhanced Mobile Broadband
EPC	Evolved Packet Core
FDD	Frequency Division Duplex
FFT	Fast Fourier Transform
FR1	NR Frequency Range 1 (below 6GHz)
FR2	NR Frequency Range 2 (above 6GHz)
IAB	Integrated Access and Backhauling
IFFT	Inverse Fast Fourier Transform
IIoT	Industrial IoT
IoT	Internet of Things
ISI	Inter-symbol interference
ITS	Inteligent Transport Systems
ITU	International Telecommunication Union
KPI	Key Performance Indicator
LTE-M	LTE for Cat-M devices
LTE	Long Term Evolution

M2M	Machine to machine
MBB	Mobile Broadband
MCS	Modulation and Coding Scheme
MIMO	Multiple Input Multiple Output
mMTC	Massive Machine type Communications
MU-MIMO	Multiple User MIMO
NB-IoT	Narrow Band IoT
NB	Narrow Band
NFV	Network Function Virtualization
NR	New Radio
NSA	Non Stand Alone
OFDMA	Orthogonal frequency-division multiple access
PAPR	Peak to Average Power Ratio
PBCH	Physical Bradcast Channel
PCC	Primary Component Carrier
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PF	Proportional Fair
PRB	Physical Resource Block
PSS	Primary Synchronization Signal
QoS	Quality of Service
RAN	Radio Access Network
RB	Resource Block
RBG	Resource Block Group
RR	Round Robin
RRC	Radio Resource Control
SA	Stand Alone
SC-FDMA	Single Carrier frequency-division multiple access
SCC	Secondary Component Carrier
SCS	Sub carrier Spacing
SDN	Software Defined Networks
SINR	signal-to-interference-plus-noise ratio
SRB	Signalling Radio Bearer
SSS	Secondary Synchronization Signal
SU-MIMO	Single User MIMO
TB	Transport Block
TBS	Transport Block Size
TDD	Time Division Duplex
TRS	Tracking Reference Signal
TTI	Transmission Time Interval
UE	User Equipment
UL	Uplink
UP	User Plane
UPF	User Plane Function
URLLC	Ultra Reliable and Low Latency Communications
V2X	Vehicle to Anything

Esta es la última página.
Compilado el Tuesday 16th November, 2021.
http://iie.fing.edu.uy/