Tampere University

Sebastian Villegas

FRONTHAUL MODELLING FOR 5G SYSTEMS

Information Technology and Communication Sciences
Master of Science Thesis
August 2021

ABSTRACT

Sebastian Villegas: Fronthaul Modelling For 5G Systems
Master of Science Thesis
Tampere University
Wireless Communications & RF Systems
August 2021

5G networks aim to provide reliable, fast and service orientated communications to its users. One of its most attractive features, is its random access network (RAN) virtualization technologies for different vertical industries that can tailor the spectrum allocation based on the performance needs of the customers. This service pliancy brings new architectural challenges that are still being tackled both in the industry and academia. One of such, is finding a reliable, fast, flexible and cost effective fronthaul communication transport protocol via fiber optic links. Current network architectures favor Ethernet as a transport mechanism to meet the demands of fronthaul transport due to its upgradability to higher data rates, network switching, scalability and wide availability in the market. Furthermore, eCPRI and oRAN provide a set of layers that define encapsulating procedures for user and control plane data that is carried inside Ethernet frames.

This thesis work aims to build a model of a Nokia 5G subsystem IP that processes eCPRI and oRAN encapsulated IQ samples and is transported through Ethernet. The model was aimed to replicate the same processing procedures, configuration and register definitions as is done in the real implementation in the ASIC/FPGA. The motivation behind building the model is to recreate the ideal behaviour of the subsystem in order to have a tool that allows system designers and architects to test new features and algorithms, which can greatly reduce costs and time, and also provide verification engineers with a tool to generate ideal responses of the system when testing the real physical implementation of the system. Furthermore, the model can be integrated with other available subsystem IP models and thus have an orchestrated ideal behaviour of a complete system module.

Keywords: Ethernet, 5G, Telecommunications, eCPRI, oRAN, Fronthaul, RAN

PREFACE

The research work presented in this master thesis has been carried out in Nokia Networks, Finland. The work was supervised in Nokia by Arto Palin and in Tampere University of Technology by professor Jukka Talvitie.

I would like to thank my supervisors, Arto Palin and Jukka Talvitie, for guiding me throughout this thesis work and for their guidance in writing this document. I am grateful to Juha-Matti Saarinen and Tero Joentakanen, for always answering my inquiries, and to my line manager Jyrki Hyrsylä, for always supporting me. Furthermore, I would like to extend my gratitude to all my friends and colleagues which helped and guided me throughout the completion of this work.

Lastly, I would like to thank my family and specially my mother Nora, without whose unconditional love and support I would not have been able to carry on.

Tampere, 16th August 2021

Sebastian Villegas

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

5GC	5G core network
BWA	bandwidth parts
CPRI	common public radio interface
CU	central unit
DU	distributed unit
eCPRI	enhanced common public radio interface
gNB	next generation nodeB
gNB-CU	gNB central unit
gNB-CU-CP	gNB-CU control plane
gNB-CU-UP	gNB-CU user plane
gNB-DU	gNB distributed unit
IQ	In-Phase and Quadrature components
IWF	eCPRI/CPRI Interworking Function
LAN	local area network
MIMO	multiple input multiple output
NFV	network function virtualization
NR	NR radio access
OFDM	orthogonal frequency division multiple access
PCH	paging channel
PCI	physical cell identifier
PDCCH	physical downlink control channel
PDSCH	physical downlink shared channel
PDU	protocol Data Unit
PHY	physical layer
PRACH	physical random access channel
PRB	physical resource block
PUCCH	physical uplink control channel
PUSCH	physical uplink shared channel
QAM	quadrature amplitude modulation

QPSK	quadrature phase shift keying
RACH	random access channel
RE	radio equipment
REC	radio equipment control
SCS	subcarrier spacing
SDN	software defined network
UE	user Equipment
Xn-C	Xn-control plane
Xn-U	Xn-user plane

1 INTRODUCTION

Robots, self driving cars and smart cities are no longer science fiction. Services that were unthinkable decades ago - smart energy, autonomous operated industrial machines, drones, self driving cars and virtual health - became integral part of our present day lives. Global technological advancements are growing swiftly and with them the need for a fast, reliable and flexible communication network. 5G NR aims to cover emerging telecommunication demands in a way that empowers radio and network design to evolve in a natural and progressive way in order to meet current and future challenges. Some of the system design solutions include flexible deployment and virtualization technologies that provide 5G the necessary tools to support service-orientated architectures and interfaces tailored to the end user's needs. Unlike LTE, 5G networks are thought of to satisfy different communication requirements depending on the specific demands of its customer. Industrial automation machines, for example, rely on communication that has to be reliable and low latency but does not consume great amount of bandwidth, whereas gaming platforms or 4K streaming services rely more on high bandwidth and speed. Moreover, IoT systems - such as smart cities - require the ability of the network to support hundreds of devices connected at the same time and to the same network so that the "things" can interact with each other. Throughput (bandwidth) and low latency are not generally prime concerns of IoT system but massive connectivity is. 5G sets out to cover the needs for high bandwidth, massive connectivity and ultra reliable communication within the same network implementing the same hardware. In other words, the same network infrastructure will provide a tailored service to each of the connected users to provide the specific communication requirements of that user. For example, if a factory that contains robotic arms for processing is connected to the same network as a residential building, the network can allocate ultra-reliable time critical communication services to the factory and high bandwidth and fast communication to the residential building, all within the same network. The new infrastructure has identified the aforementioned needs as three use cases to tackle : enhanced mobile broadband (eMBB), massive machine-type communication (mMTC) and ultra-reliable and low-latency communication (URLLC). Although these three cases have been identified, there are some cases that do not fall into any particular case but rather require a combination of two, which is also supported. Health IoT applications and satellite positioning require the support of massive amounts of sensors that require high bandwidth as well as ultra reliable communication, so both categories fall in between two of the three use cases. Figure. 1.1 illustrates the three basic use cases and some of its possible applications including some applications that require shared capabilities between two use

cases.

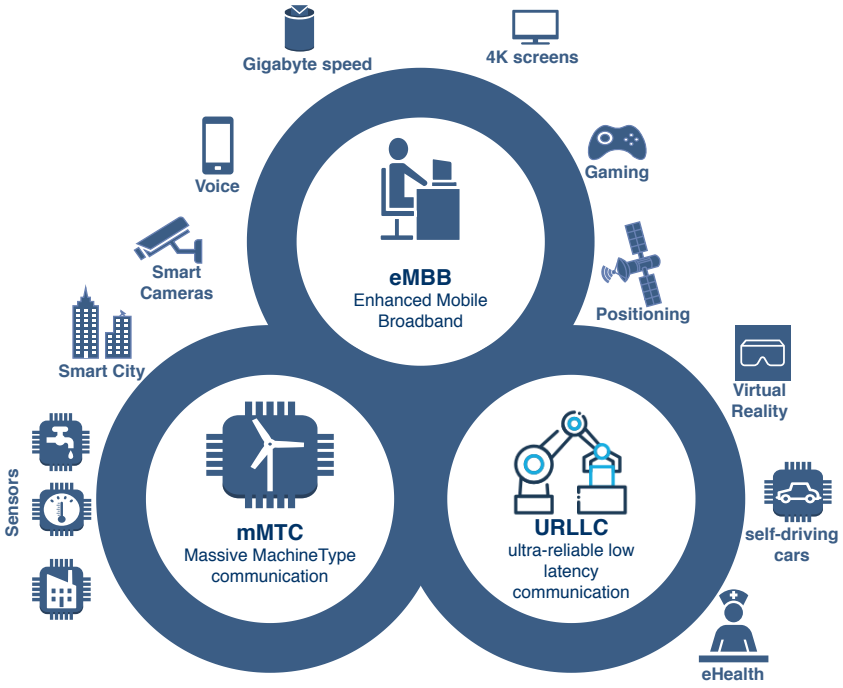


Figure 1.1. *5G Targeted Use Cases*

The network's architecture design faces big challenges that are introduced by tailoring the communication networks to specific markets. In order to combat such challenges, the RAN architecture is adapting new emerging technologies that can provide ease of deployment and upgradability. One example is the RAN including software and cloud network components to have a bigger presence than before. This makes network updates and scalability a lot easier since there is no need to physically change anything to perform updates or to change the network. Furthermore, software and cloud network components also allow changes in functionality to take place faster, since software development can be quicker than hardware development, and at a lower price. Moreover, mobile operators now are pressured to carry on the new services that come with 5G in an agile way while maintaining the prices low.

5G's diverse nature requires a flexible interface among different components of the network. Received signals containing user data can be transported through fiber optic links into different network components so that they can be processed and routed to the 5G network core. The new architecture divides different processing layers into different physical units, as opposed to LTE which had one centralized baseband unit. This separation created a communication link called fronthaul, which is a physical connection between a remote radio head - containing antennas and RF processing - to a processing unit called the distributed unit - that processes data and then routes it to another processing unit called the control unit. Fronthaul transport must occur in such a way that the timing, quality of service (QoS) and speed of the communication link is not compromised or the data distorted while meeting strict network timing requirements and maintaining low prices. LTE used transport mechanisms that require expensive hardware components im-

plementing proprietary protocols and that had complex mechanisms with stringent time requirements. 5G looks to implement mechanisms that are open, widely available and reduced in complexity while meeting network requirements. Ethernet and IP/UDP are transport mechanisms that are readily available in the market and have been considered as solutions for fronthaul connectivity. However, Ethernet's ability to evolve to meet new requirements and solve challenges in the changing demands and needs of networks, as well its scalability, upgradability to higher data rates, high speed optical connections and cost efficiency allow it to be a prime candidate protocol to embrace the new challenges that come with 5G's fronthaul connectivity.

Fronthaul transport is a hot topic of discussion in 5G NR and several solutions are being explored by researchers and telecommunication companies. One of the proposed solutions and the solution followed in this thesis is to encapsulate the data following the enhanced common public radio interface (eCPRI) and oRAN protocols and carrying it in the fronthaul using Ethernet frames. Ethernet provides a well established technology that is easily accessible and has an open standard. Furthermore, Ethernet has been around for many years and continuously adapts to new transport demands by implementing new algorithms and changing its hardware requirements. Being an open standard brings portability, flexibility and helps to maintain development prices down. The transported data is encapsulated inside a protocol layer called eCPRI and it provides functionality headers that are related to the data that is inside the Ethernet frame. eCPRI alleviates transport challenges that arise due to new antenna functionalities. For example, in the case of implementing multiple antennas to transmit and receive (MIMO) where selected reception algorithms (beamforming) is shifted to the remote radio heads. By using eCPRI (assuming digital beamforming), the increase of the data rate is proportional to the number of antennas [15]. Moreover, an open RAN architecture, proposed by the O-RAN alliance, separates software and hardware RAN components and creates an interface between them, allowing software orientated processing to perform RAN processing. Finally, oRAN specifies how the IQ and control data should be wrapped inside the eCPRI layer to be carried by the Ethernet frames.

1.1 Objective and Scope of Research

Data packets containing user and control data are transmitted and received between remote radio heads that are placed in different locations of the network and a distributed unit. When electromagnetic waves carrying user equipment (UE) signals are received in the antennas of the radio heads, the RF and lower part of the physical layer processes the data, encapsulates it and sends it through a fiber optic link as an Ethernet frame to the upper part of the physical layer that is found inside a distributed unit physically placed in a different location. Conversely, when data comes from higher processing layers, the information is encapsulated and sent through a fiber optic link from the upper physical layer to the lower physical layer and RF DFE (digital front end). Frames containing control data are sent first so that the processing unit knows specific details of the user IQ

samples that are transported following the control packets.

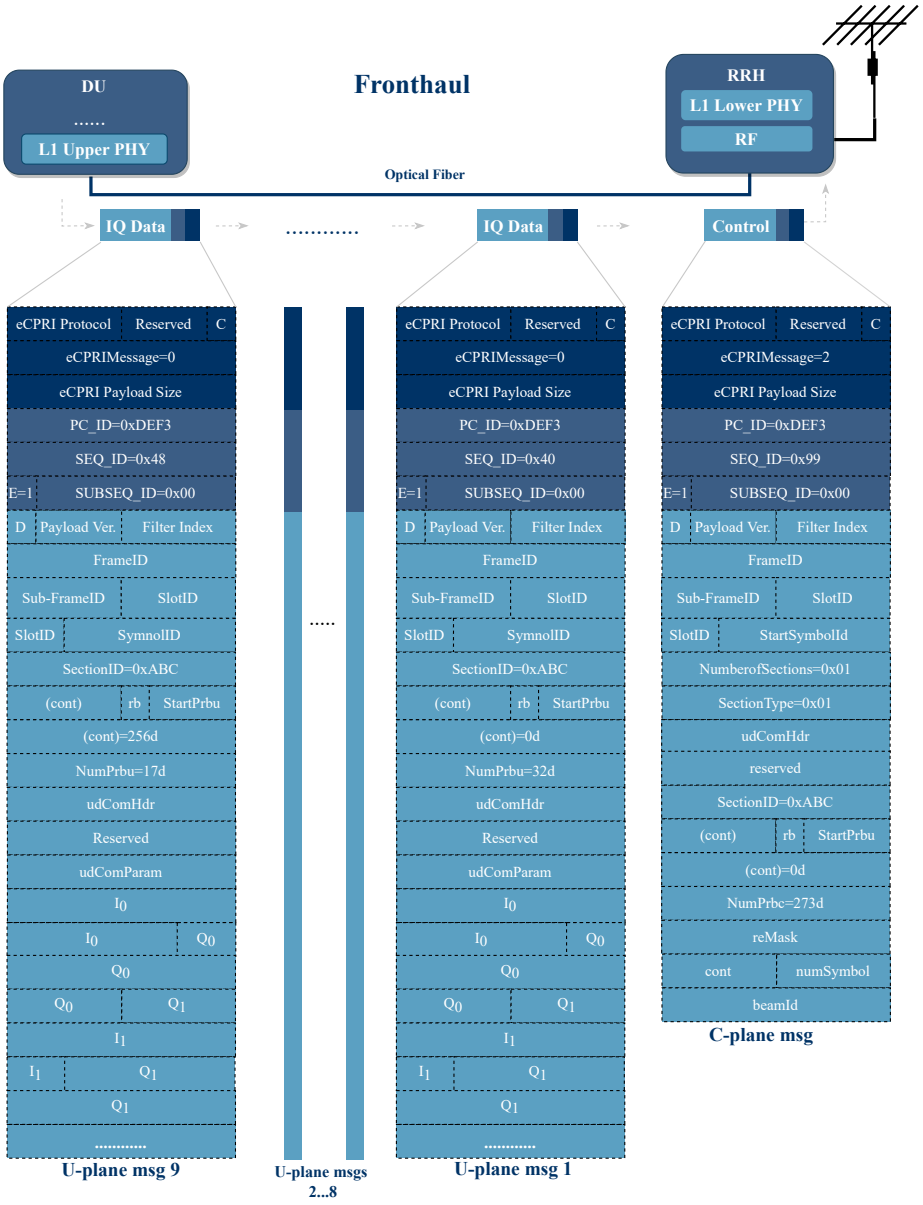


Figure 1.2. *Fronthaul transmission via optical links using eCPRI and oRAN encapsulated frames over Ethernet.*

Figure. 1.2 shows an example of downlink transmission between the DU and an RRH carrying one control frame and a couple of IQ data frames. The frame structure encapsulated inside both types is also shown in the figure with example values. The objective of this work was to create a model of a Nokia 5G subsystem IP that processes eCPRI and oRAN encapsulated IQ samples and that is transported through Ethernet. The model was aimed to replicate the same processing procedures, configuration and register definitions as is done in the real implementation in the ASIC/FPGA. Figure. 1.2 presents a transmission scenario that resembles the target scenario for the model to execute. The model focuses on the ideal behaviour of the IP when it comes to processing the frames, parsing its data and feeding it to the next processing module of the system. Moreover, the IQ samples carried in the frames are written to AXI addresses and the rest information related to the

IQ sample are passed to the following processing modules in the system. The Ethernet model can be integrated to different subsystem models to recreate the ideal behaviour of a complete processing module. This allows design engineers and system architects to change system features and see an ideal response to such changes almost instantly instead of synthesizing them into ASIC/FPGA and testing them, which takes a considerable amount of time. Furthermore, the model parsed data can be used as test vectors for verification engineers to compare against the results of the real system and determine if the system is behaving properly. An overview of the system and the modelling flow will be presented in Chapter 3.

1.2 Thesis Structure

The outline of this thesis work is as follows. First, Chapter 2 introduces the theoretical foundation for fronthaul communication in the 5G RAN and some of the protocols that are implemented in the encapsulating data. It also discusses different transport protocols that are explored in the available literature and that will be implemented in the transport model. Chapter 3 introduces the behavior of the IP in a general view, the modelling workflow that was used and the results from the model. Lastly, Chapter 4 concludes the thesis and discusses future proceedings.

2 BACKGROUND AND LITERATURE REVIEW

This chapter introduces some of 5G RAN's features and highlights their importance for fronthaul transport. It also presents the theoretical foundation for the Ethernet, eCPRI and oRAN protocols. First, Section 2.1 introduces 5G's functional split - with its different options and use cases- and mixed numerology. Then, Section 2.2 provides an overview and explains how the Ethernet protocol works and why it is a good candidate to send control and user information in fronthaul links. Finally, Section 2.3 and Section 2.4 cover the eCPRI and oRAN standards for encapsulating data.

2.1 5G RAN

5G is designed to support a wide range of diverse services with different data traffic needs such as high throughput, low latency, IP traffic, etc. One of its main characteristic is the introduction of a new radio interface (NR) that offers the flexibility needed to support those types of services. Technologies such as Software-defined networks (SDN) and network function virtualization (NFV) provide the necessary foundations for 5G to be able to implement dynamic network management concepts such as Network Slicing (NS) and mixed-numerology. It also facilitates service-orientated architectures and interfaces tailored to the user's needs. The 5G network supports two deployment options: SA (stand alone) architecture, which connects the 5G access network to the 5G core network, and NSA (non-stand alone) architecture which connects the 5G RAN to 4G(LTE) core network. The co-located deployment of LTE eNBs and NR gNBs using non-stand alone deployment will dominate in the initial stages of 5G and later the stand alone deployments will become more predominant [4].

As discussed in the introduction and shown in Fig. 1.1, network use cases have emerged naturally as requirements for 5G networks: massive machine type communication (mMTC), intended for IoT type of scenarios (sensors,actuators,etc); ultra-reliable and low latency communication (URLLC), used industrial automation and control; and enhanced mobile broadband (eMBB), used in regular user type of scenarios such as gaming and internet browsing. These use cases provide a challenge for resource management when it comes to the electromagnetic spectrum because they are intended to be used at the same time in the same network. In order to facilitate such spectrum allocation, mixed-numerology is introduced to 5G networks to provide customized resources per application or intended use.

This topic will be discussed in section 3.3.2. Two other prominent ideas in 5G networks are MIMO (multiple input multiple output) and beamforming. MIMO increases the throughput and capacity of a channel by using multiple antennas to transmit and receive signals and beamforming is a technique that focuses the antenna's beam to a specific device in order to increase its antenna's directivity and attenuate interfering signals. The specific details of MIMO and beamforming are outside the scope of this thesis. However, they are an important factor in fronthaul transport since every antenna port is considered an independent stream of data that is either transmitted or received. Stream separation has to be considered and included in the eCPRI and oRAN encapsulation when transferring the data. The beamforming coefficients and weights are also transported in specified fields in control frames, such as the one observed in Fig. 1.2.

2.1.1 Functional Split

The 5G RAN architecture presents physical changes such as the separation of processing units between functions that are closer to the radio or that are asynchronous to the radio. It creates the separation by disaggregation what in LTE is known as the base station into two entities called the central unit (CU) and the distributed unit (DU) and it also defines a separated data flow for the control plane (CP) and the user plane (UP) information. This new implementation greatly differs from that of 4G which had a clearly defined group of entities performing statically determined functions. This limitation is noticeably improved in the 5G physical architecture since they are a lot more flexible in terms of physical network entities. According to [1], the principles of the NG-RAN (new generation RAN) architecture are: to achieve a logical separation of signalling and data transport, to separate the NG-RAN, 5GC functions and for an RRC connection to be fully controlled by the NG-RAN, to have the least amount of divisions across interfaces, to have one physical network element that can implement multiple logical nodes and to have entities that are controlled by interfaces which are based on a logical model.

The architecture of the NG-RAN consist of several network elements called gNBs and ng-eNBs that are connected with each other, through an X_n interface, and also connected to the 5G core (5GC), as seen on Fig 2.1. More specifically, they are routed inside the 5GC to the the Access and Mobility Management Function(AMF) via a NG-C interface and User Plane Function (UPF) with a NG-U interface. Some of the gNB and ng-eNB functions are: radio resource management, IP header compression, encryption and protection of data, routing user and control plane data towards UPFs or AMFs, connection setup and release, scheduling and transmission of broadcast and paging, measurement reporting, radio access network sharing,etc. Nonetheless, they both have specific scenarios: a gNB is a node that gives the user and control plane a protocol towards the user equipment(UE), whereas a ng-eNB is a node that provides e-UTRA user and control plane protocols towards the UE. Both gNBs and ng-eNBs are routed to the 5GC with a NG interface. Moreover, A single gNB may consist of a gNB-CU-CP, multiple gNB-CU-UPs and multiple gNB-DUs [4].

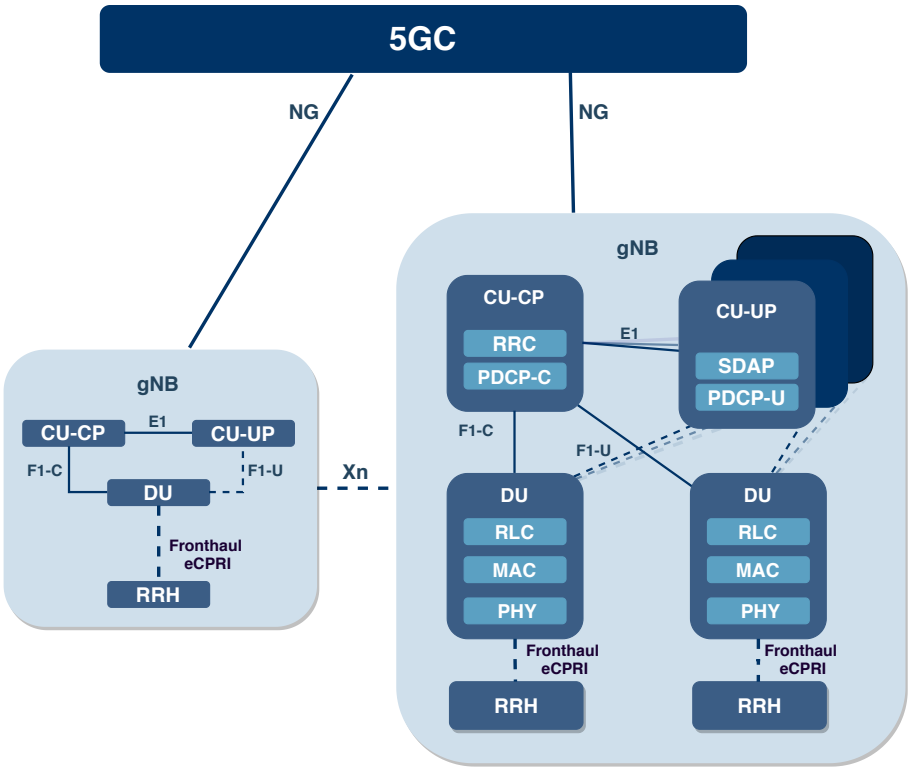


Figure 2.1. *gNB architecture as described in [1] and [3]*

The gNB-CU-CP is connected to the gNB-DU through the F1-C interface and The gNB-CU-UP is connected to the gNB-DU through the F1-U interface. However, one gNB-DU can only be connected one gNB-CU-CP and one gNB-CU-UP can only be connected to one gNB-CU-CP [1]. There are two deployment options for physically placing the central unit and distributed unit that are considered[2]: Collapsed gNB deployment, disaggregated gNB deployment. In the collapsed deployment, all RAN functions are located in the same site, as it occurs in LTE systems.

In the disaggregated deployment, the RAN functions are distributed across different sites. The DU and CU can be physically apart and the CU is divided into CU-CP, containing the PDCP-C and RRC, and CU-UP, which hosting the PDCP-U and SDAP. The DU carries the RLC, MAC and PHY protocols. The separation of the control plane and user plane entities allows for a greater optimization based on topology and performance. A new interface called E_1 is used to coordinate the CU-CP and CU-UP entities although traffic is not exchange between both entities.

3GPP conducted several studies on different possible options for a functional split between the CU and DU to determine differences, advantages and disadvantages of placing different processing units in different locations of the RAN. Furthermore, the study examined how placing processing units of different protocol layers can have different responses to factors such as load demand, user density, transport options, QoS, latency, high throughput, etc. The benefits sought in partitioning the functionalities between both entities were to create

a flexible functional architecture that include, according to [4]: Flexible hardware implementations that bring cost effective solutions, coordination for performance features, load management, real-time performance optimization, enabling NFV/SDN and adaptation of variable latency on transport. Figure. 2.2a shows the possible options for the separation of the processing units in the RAN.

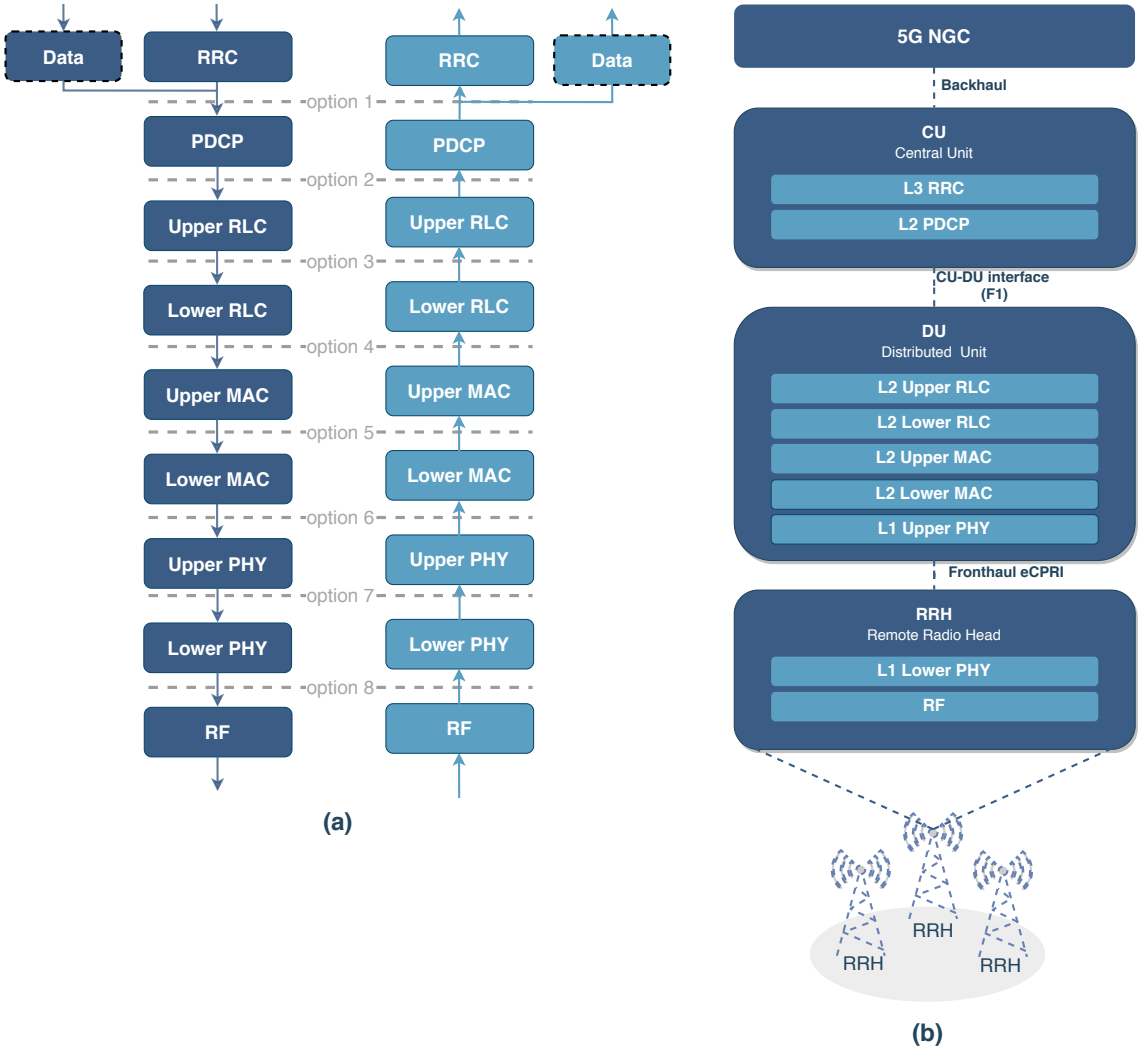


Figure 2.2. *Functional split between the CU and DU as described in [4]*

The specific functionality of the layer is directly controlled by the separation option that is chosen. Some layers can be separated into a lower and higher processing unit of the same OSI layer protocol. For instance, if option 5 is chosen, it creates an intra-MAC layer split that constitutes a separated lower and upper MAC processing units that are placed in different locations of the RAN. In such case, the RF, physical layer and lower part of the MAC layer are placed in the distributed unit and the higher part of the MAC layer, RLC and PDCP in the central unit. 3GPP determined that options 2 and 7 are the most balanced overall, although no agreement on the internal split for option 7 has been reached [5]. Fig. 2.2b presents an example where options 2 and 7 are implemented, where the DU contains the RLC, MAC and the upper PHY layers, the CU hosts the RRC and PDCP

and the RRH has the lower PHY and RF components.

The intra-PHY division seen in option 7 has three possibilities, some which provide asymmetrical options to benefit uplink and downlink transmission independently. They are option 7-1, option 7-2 and option 7-3. Option 7-1 allocates the FFT processing, CP and optional PRACH filtering inside the DU for uplink transmission and the iFFT processing and CP for downlink transmission. Option 7-2 deploys the FFT processing, CP removal, resource de-mapping and optional pre-filtering processing inside the DU for uplink transmission and the iFFT processing, CP addition, resource mapping and pre-coding for downlink transmission. All the remaining PHY functions for options 7-1 and 7-2 are placed inside the CU in both uplink and downlink directions. Finally, option 7-3 allocates the encoder inside the CU and the rest of the processing units goes inside the DU. This option is only present for downlink.

The aforementioned options intend to allow the traffic transmission points to be centralized, relax the fronthaul requirements, centralize scheduling and perform joint processing in both transmitting and receiving (since the MAC is inside the CU). However, a possible set back in separating processing units is that the split might require sub-frame level timing interactions between parts of the PHY in the central unit and the distributed unit [4]. Nonetheless, the processing layers that are divided in the RAN are still capable of processing 5G features such as MIMO, beamforming, mixed-numerology, etc. This is specially relevant to achieve the frequency sharing and multiplexing that is needed for supporting different vertical industries. Different options for intra-PHY splitting will be revisited when the eCPRI transport protocol is introduced in Sec. 2.3 and once again when the O-RAN proposed data encapsulation is presented in Sec. 2.4. The following section provides an an introduction to mixed-numerology and discusses its fronthaul challenges.

2.1.2 Mixed Numerology

5G NR introduces OFDM numerologies for different frequency band requirements of the three deployment cases. Numerology can be defined as the parametrization of a physical waveform [23]. Contrary to LTE, the different numerologies can be mixed and used simultaneously in the same 5G network and can be implemented dynamically; this concept is called mixed-numerology. The OFDM spacing used is determined based on channel factors that can compromise the quality of the signal such as the delay spread, Doppler spread, phase noise and carrier frequency. The subcarrier spacing (SCS) is the span in frequency of a single carrier and is the parameter that determines the specific numerology for a single carrier. The sub-carrier spacing is a parameter that is represented by scaling a basic spacing of 15kHz with 2^{μ}, where μ is a positive integer that is defined to span

integers from 0 to 4. A subcarrier-spacing can be given as

$$\Delta f = 2^{\mu} \cdot 15\text{kHz} \tag{2.1}$$

and consequently, the OFDM symbol duration is determined as

$$T_u = \frac{1}{\Delta f} \tag{2.2}$$

The useful OFDM timing is inversely proportional to the SCS (Eq. (2.2)), that is, the bigger the frequency span, the shorted the symbol duration is making these two parameters are a trade off between latency and spectrum allocation. However, lower useful symbol times are affected more severely by channel distortion. In order to mitigate the damage caused by the channel, OFDM symbols append a cyclic prefix or guard interval with redundant data in the beginning of the wave. This makes the symbols more robust to timing synchronization errors and channel distortions due to different scenarios such as Doppler and multipath. Cyclic prefixes usually have a chosen length T_{cp} to be less than or equal to $T_u/4$ and it will ideally be greater than the channel RMS delay spread (τ_{rms}). The OFDM symbol duration then becomes the addition of the useful symbol length and the length of the cyclic prefix as

$$T_s = T_u + T_{cp} \tag{2.3}$$

where the slot length can be defined as

$$slot\ length = \frac{1\text{ ms}}{2^{\mu}} \tag{2.4}$$

The length however changes depending on the scaling factor of the SCS.

The frame structure for every subcarrier sets the time and physical characteristics of the signal transmission. The maximum subcarrier spacing (Δf_{max}) is 480kHz with an FFT size (N_{fft}) of 4096. A basic time unit T_c is used to create consistent and exact time intervals for specifications in NR. It can be interpreted as a sampling time of an FFT based transmitter or receiver using an FFT size of 4096 of a frequency of 480 kHz. The sampling time (T_c) expression can be defined as

$$T_c = \frac{1}{\Delta f_{max} N_{fft}} \tag{2.5}$$

Frames carrying uplink and downlink messages are made up of sub-frames that have slots inside. Each slot carries OFDM symbols with the transmission data. 5G has a fixed frame and sub-frame duration regardless of the subcarrier spacing implemented. However, the amount of OFDM symbols and slots vary depending on the configuration. Each frame has

time domain length of

$$T_{frame} = \left(\frac{\Delta f_{max} N_{fft}}{100}\right) T_c = \left(\frac{\Delta f_{max} N_{fft}}{100}\right)\left(\frac{1}{\Delta f_{max} N_{fft}}\right) = \left(\frac{1}{100}\right) = 10\text{ms}$$

and each sub-frame inside a frame has a duration of

$$T_{sub-frame} = \left(\frac{\Delta f_{max} N_{fft}}{100}\right) T_c = \left(\frac{\Delta f_{max} N_{fft}}{100}\right)\left(\frac{1}{\Delta f_{max} N_{fft}}\right) = \left(\frac{1}{1000}\right) = 1\text{ms}$$

Thus, each frame is divided into 10 sub-frames.

The number of OFDM symbols per sub-frame depends on the sub-carrier spacing configuration μ. The number of symbols per frame (N^{symb}_{frame}) is equal to the number of slots per sub-frame ($N^{slot}_{subframe}$) multiplied by the number of symbols per slot (N^{symb}_{slot}).There are generally 14 continuous OFDM symbols per slot although that depends on the length of the CP. Table. 2.1 presents some of the numerical definitions according to what spacing configuration is used.

Table 2.1. *Mixed-numerology parameters*

μ	cp	Δf [kHz]	N^{slot}_{symb}	$N^{subframe}_{slot}$	N^{frame}_{slot}	T_u [μs]	T_{cp} [μs]	T_s [μs]
0	Normal	15	14	1	10	66.67	4.7	71.36
1	Normal	30	14	2	20	33.33	2.34	35.68
2	Normal	60	14	4	40	16.67	1.17	17.84
2	Extended	60	12	4	40	16.67	4.17	20.84
3	Normal	120	14	8	80	8.33	0.57	8.92
4	Normal	240	14	16	160	4.17	0.29	4.46

Resource elements are the smallest physical resource in NR and consist of one subcarrier during one OFDM symbol. A collection of 12 consecutive subcarriers in frequency domain are called a resource block. Both are illustrated in Fig. 2.3, where k is the index in frequency domain and l is the symbol position in time domain. Resource blocks are defined as one dimensional measures that span only in frequency domain to provide flexibility in the time duration for different transmissions, as opposed to LTE which used two dimensional resource blocks that spanned 12 subcarriers in frequency domain and one slot in time domain [11].

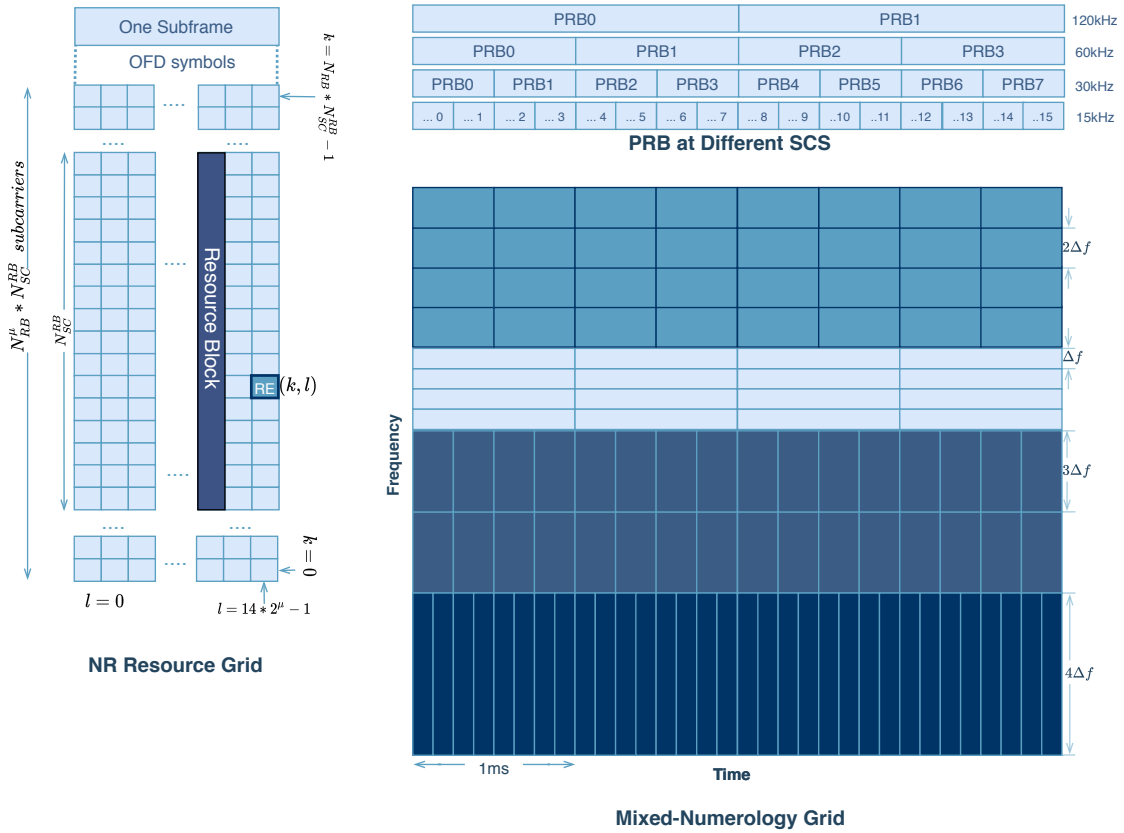


Figure 2.3. *NR resource grid in different mixed-numerology scenarios and with different PRB divisions.*

The frequency span (in Hertz) varies for a resource block (12 subcarriers) depending on the spacing configuration μ since different SCS are implemented in the same network. For example, in the case of a subcarrier spacing of 15kHz, a resource block spans to 180kHz, whereas a spacing of 120kHz would require 1.440MHz of bandwidth for a single resource block. Physical resource blocks (PRB), which are used to describe the physically transmitted signals, span 12 subcarriers that have the same subcarrier spacing and CP length and are multiplexed in time domain when multiple numerologies are used. PRBs are all aligned to multiples of 15kHz to keep backwards compatibility with LTE systems and to have a systematic way of dividing the spectrum. A resource grid of $N_{RB}^{max} * N_{SC}^{RB}$ subcarriers and $N_{subframe}^{symb}$ OFDM symbols is defined for every carrier and numerology.

Fig. 2.3 shows an example of a NR resource grid as well as the the different length of PRBs depending on the subcarrier spacing. Moreover, Fig. 2.3 also shows a mixed-numerology grid example where 4 different SCS are used in the same grid. Numerology values are used depending on the specific application and frequency/time grid demands of the network. Larger sub-carrier spacing allows less latency - since the symbol duration is lower - and shorter spacing allows for a larger cell size which is needed in lower frequency scenarios. Moreover, smaller sub-carrier spacing can be heavily affected in high Doppler scenarios. They require CP overhead and anticipated delay spread to set an upper limit for sub-carrier spacing and guaranteed robustness against phase noise and Doppler spread are used to

set the lower limit. However, large sub-carrier spacing can add unwanted overhead. For instance, time critical applications could benefit from employing a wider SCS since the primary focus is low latency, and IoT applications - that are not time critical- can have a more relaxed time span and employ a smaller SCS. Fig. 2.3 portrays different scenarios that can fulfill aforementioned requirements on a NR resource grid with different SCSs.

Mixed-numerology plays a key role in the design of 5G systems and brings challenges that were never dealt with in older telecommunication architectures. It allows an optimal resource allocation that enables the network to provide service-orientated communications for the end users. It also introduces obstacles in fronthaul transport since the payload is no longer uniform with statically configured parameters so the transport mechanism must be able to store dynamic parameters that belong to the transported data so that the receiving unit can process and route the information properly.

Transport schemes, such as CPRI, have been implemented in LTE systems for carrying IQ data and its new evolution, eCPRI, promises to meet the flexibility requirements needed by fronthaul transport in NR. Furthermore, Ethernet stands in the spotlight as a generic, cost effective, widely known and scalable method to carry eCPRI and oRAN protocol payloads, specially because it has the possibility of carrying eCPRI frames in a switched network that can carry real-time control information and user data. It is expected that fronthaul networks will become more complex multi-hop topologies which require switching and aggregation.

2.2 Ethernet

Ethernet is a protocol used to interconnect devices inside a local area network (LAN). It has been used to connect internet to homes and workplaces through a series of evolutions since it was first introduced in 1972 by Bob Metcalfe in Xerox Palo Alto Research Center. Its flexibility, scalability, adaptability and reliability has made it to be a widely used networking technology which can be found everywhere [30]. Ethernet was first inspired in the Aloha network protocol but it was then improved by using collision detection, carrier sensing and multiple access transmission support. The first standard for 10Mb/s Ethernet was written in 1980 by Digital Equipment Corporation, Intel and Xerox consortium; the standard was open and thus became the world's first open, multi-vendor LAN standard to which - years later - Metcalfe quoted: "The invention of the Ethernet as an open, non-proprietary, industry-standard local network was perhaps even more significant than the invention of Ethernet technology itself" [30].

Ethernet has seen a significant amount of changes in speed as the physical mediums have been improved over the years. The very limited coaxial Ethernet system saw a great improvement on speed and reliability with the introduction of twisted-pair Ethernet in the late 1980s. The twisted-pair allowed the transmission speed to be increased from 10 Mb/s to 100Mb/s at first and later to 1000Mb/s, introducing the first gigabit Ethernet. Auto-negotiation protocols were also introduced so that the Ethernet interface can configure the transmission speed to be the highest supported link speed by switching to ports that support full-duplex mode. In 2003, the 10Gb/s Ethernet standard publication defined a set of fiber optic media systems that, although it does not support the original CSMA/CD (Carrier Sense Multiple Access with Collision Detection) of its older versions, can now operate in full-duplex mode. In recent years, Ethernet signals over fiber optic cables and short copper coaxial cables can reach speeds up to 40 and 100 Gb/s. The vast improvements that Ethernet has seen over the years are not only in regards to speed but also on its capabilities, such as power over Ethernet(PoE), auto-negotiation and full-duplex to several connected devices. Ethernet is a technology that is consider a very strong solution to the fronthaul transport challenges because it is widely present and available within access networks. Ethernet has advantages such as low cost of equipment, easy upgradability to higher data rates, ease of scalability, high speed optical networks, sharing equipment with fixed access, statistical multiplexing gains, network switching, routing and monitoring hardware. Furthermore, Ethernet based fronthaul is an attractive transport technology solution due to its reduction in jitter, which can be as much as 1 μs [7] and it also allows legacy equipment such as CPRI to be used to operate at the edge of a fronthaul network [15].

An Ethernet network works by combining hardware and software components. There are four basic elements that make up an Ethernet subsystem: the frame, the media access control protocol (MAC), the signaling components and the physical medium. A frame is a standardized set of bits arranged in a specific way that carries data over a system, the MAC protocol is a set of rules on each Ethernet interface that allows the devices to

access the Ethernet channel, the signaling components are electronic devices that send and receive signals over the Ethernet channel and the physical medium is made up of cables and hardware that physically carry the Ethernet signals between the devices and the network. From a fronthaul link point of view, the most important element of Ethernet is the frame since it provides the flexibility to encapsulate the transport data with specialized protocols for telecommunications.

2.2.1 Frame Encapsulation

Ethernet frames can be of two general forms: ypye encapsulation and length encapsulation; both types can be seen in Fig. 2.4 and Fig. 2.5. Both frame structures share some common fields: preamble/SFD, destination address, source address and frame check sequence. The Preamble/SDF is the first field on an Ethernet frame and is 8 bytes wide. The preamble is made up of 7 bytes all set to a value of 0x55 and is used by the receiver to synchronize all incoming frames. The last byte, called start of the frame delimiter (SFD), holds a value of 0xD5 and it works as a mark for the receiving node to know that the destination address field will begin.

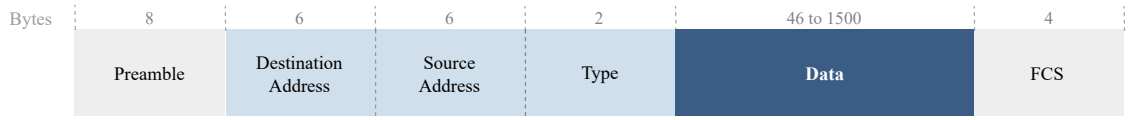


Figure 2.4. *Type encapsulation Ethernet frame*

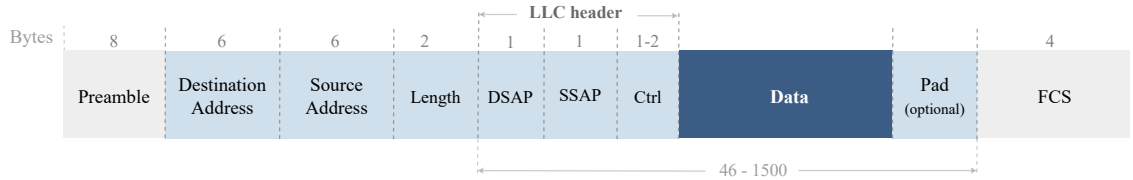


Figure 2.5. *Length encapsulation Ethernet frame*

All together, the preamble/SFD, send alternating bits (1 and 0) for 62 consecutive times except the last 2 bits which are set to 1. The following two fields in the frame, destination and source address, are both made up of 6 bytes and hold the destination address of the frame and the unicast address of the node that is transmitting the frame, respectively. In other words, the destination address tells the system where the frame must be delivered to and the source address tells the system who is sending the frame. The very last field of the frame, called frame check sequence (FCS), is a cyclic redundancy check (CRC) that is computed as a function of the contents of the fields of the frame such as destination address, source address, length/type, MAC client data and pad. According to [16], the encoding is done by the generation algorithm

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (2.6)$$

The 32 bits of the CRC are arranged so that the x^{31} term is the left most bit of the first octet and the x term is the right most of the last octet so that the bits are transmitted in descending order: $x^{31}, x^{30}, x^{29}, x^{28}, x^{27}, x^{26}...x^3, x^2, x^1, x^0$. The corresponding steps to define a CRC value corresponding to a MAC frame are specified in by the IEEE standard for Ethernet [16]. A general overview of calculating a CRC is: Complement the first 32 bits of the frame, the n bits of the protected fields are coefficients of a polynomial $M(x)$ of degree $n-1$ so the destination address field is the x^{n-1} term and the last bit of the MAC client data field is x^0. Then, $M(x)$ is multiplied by x^{32} and divided by $G(x)$ which produces a remainder $R(x)$ of degree ≤ 31. The coefficients of $R(x)$ are considered to be a 32-bit sequence which is complemented to finally obtain the CRC.

Type encapsulation frames provide fields inside the frame to specify the upper-layer protocol type and data. The type field contains the protocol running above the Ethernet so that it can multiplex among different protocols in higher layers such as IP, IPX,etc. and in the context of fronthaul transport, the eCPRI type is placed here. The data field encapsulates higher layer information. The higher layer protocol must ensure that there is at least 46 bytes of data being transported and/or have a padding mechanism to fill the field so that it satisfies the length's minimum requirements while staying below 1500 bytes. On the other hand, length encapsulation frames such as in Fig. 2.5, explain the length of the data instead of the protocol type. The two bytes after the source address field indicate the length of the payload carried in the frame instead of its upward multiplexing mechanism. The size of the data -without any padding- is specified in the length field seen in the frame structure and its able to hold from 0 to 1500 bytes or, represented in hexadecimal, values from 0x0000 to 0x05DC. The Ethernet MAC must provide any needed zero padding if there are less than the minimally required 46 data bytes. Since there is no way for the Ethernet MAC to know which higher layer protocol is being used, the frames implementing length field are assumed to encapsulate Logical Link Control (LLC) data that provides a multiplexing mechanism for different Ethernet clients.

According to [30], length and type encapsulation frames can easily coexist with each other by splitting the type/length to delegate frame type specific information. For instance, since the type/length field has 2 bytes, it can carry numerical values from 0 to 65535 and since the maximum length of frames should be 1500 bytes, then the values from 1501 to 65535 can be used as a type field. As long as the type field identifiers are assigned correctly, the values from 0x0000 to 0x05DC are reserved to describe the length field and from 0X0600 to 0xFFFF are reserved for the type field. Allowing the coexistence of both types of frames permits different Ethernet clients to communicate on the same LAN, even if both frame types can not intercommunicate, they can still share the same physical medium. Common protocols using type encapsulation are TCP/IP, IPX, DECnet Phase 4, LAT and some common protocols using length encapsulation are AppleTalk, NetBIOS, IPX(NetWare).

Ethernet is a "best effort" protocol: if bit errors occur during the transmission, the frame is dropped. It was designed this way to in order to keep the complexity of frame transmission

systems as simple and inexpensive as possible and to avoid the complexity that guaranteed reception mechanisms at the link layer have [30]. Fig. 2.6 describes in a conceptual way both transmitting (a) and receiving (b) algorithms. The main goal of the algorithm is to allow stations to decide whether the transmission is possible for them to transmit over a shared physical channel. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD) to transmit and receive the frames.

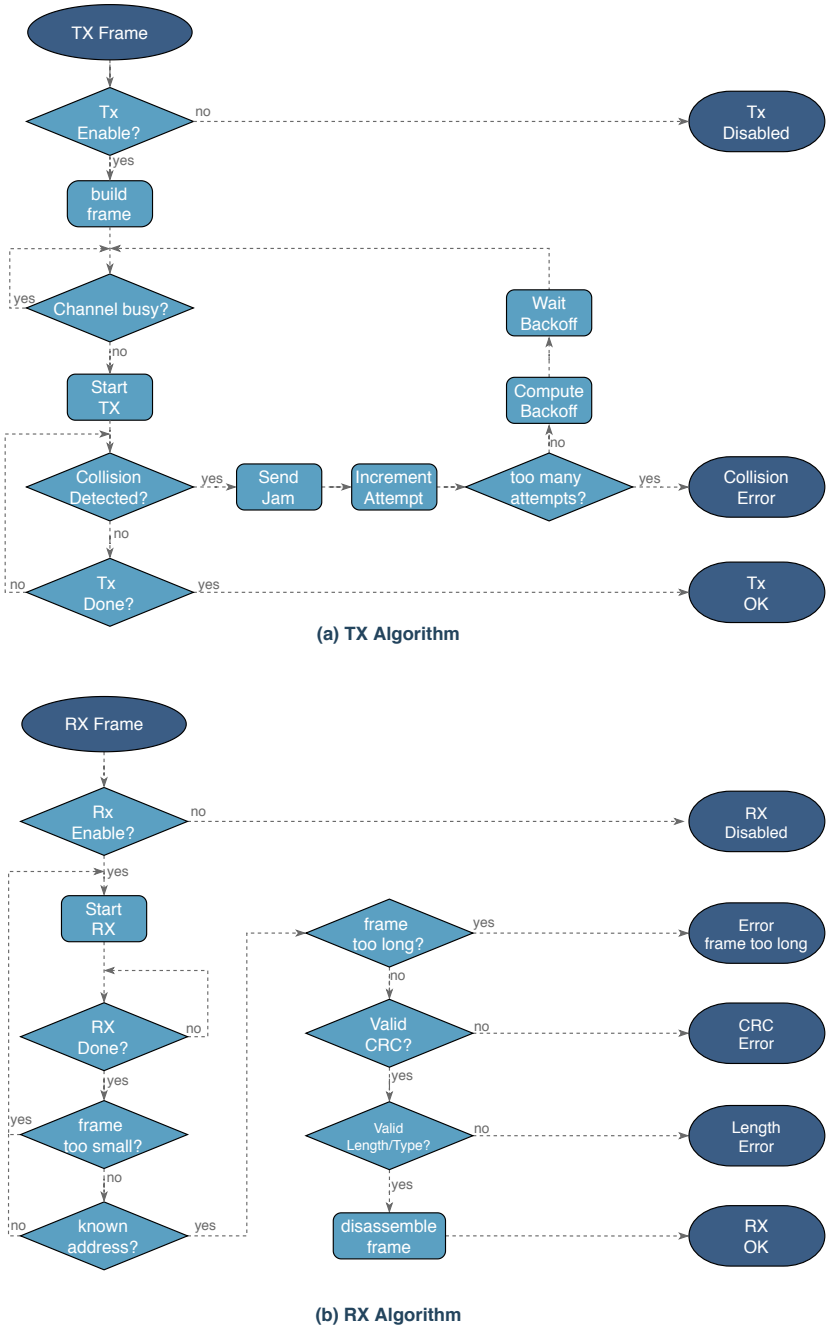


Figure 2.6. *Packet transmission algorithm as presented in the IEEE Standard for Ethernet [16]*

When a frame is queued for transmission, the transmitter module checks the physical channel to determine if its being used by any other transmitting node (sensing carrier). If the channel is occupied, the transmitter holds the transmission in order to avoid corrupting

the ongoing transmission. When the carrier is no longer sensed, the station waits for the inter-frame gap to allow the physical channel to stabilize i.e update counters, interrupting host processor, etc. After the inter-frame gap time is up, the transmitter performs the transmission according to the algorithm found in Fig. 2.6.a provided there are no more transmitting nodes attempting to transmit at the same time. If no other node interferes, the transmitter will start the transmission right after the inter-frame gap time is expired. The same procedure will be repeated if there are more frames in the queue.

Collisions occur when there are multiple transmitting nodes attempting to transmit queued frames at the same time. The collision is resolved by a collision resolution procedure that works as an arbiter to designate which station will transmit over the shared physical channel. Moreover, all nodes continue to transmit for a brief period of time to make sure that every node is aware that a collision happened; this procedure is called jamming. Once all the nodes are aware of the collision, they discard the reminder of their frames and wait for some period of time. The waiting time is selected by a random variable with exponentially increasing range that is used as a time frame before attempting retransmission. The range of the random variable r selected on the n-th transmission attempt. It starts with a range of 0 to 1, after the first collision, it increases to 0 to 3, 0 to 7,0 to 15, etc., until it reaches its maximum range of 0 to 1023. This step in the transmission scheme is called backoff procedure and is a worst-case round-trip propagation delay of the channel (slotTime). It can be represented as:

$$0 \leq r < 2^k \text{ , where } \ k = \text{Min}(n, 10)$$

The transmitting station starts the process all over after the backoff time has expired. If the frame reaches 16 collisions while attempting to transmit then: the frame is discarded, the back-off range is reset, the management counters are increased and the node moves on to transmit the following queued frame if any.

On the other end of the system, the receiver monitors the channel for a signal indicating that a frame is being received. Once the node starts to receive a frame, the preamble and start-of-frame delimiters are parsed first to synchronize and align the frame. Then the remaining of the fields in the frame are processed accordingly until the end is reached. If the frame meets the required length, the CRC is then checked to see if the frame is valid. If both fields meet the minimum requirements, the node will check the destination address and compare it with the physical address of the receiving station or if it has a multi-cast address. The MAC then feeds the frame to the corresponding device driver to process the rest of the fields and the receiving node goes back to the beginning of the receiving routine. The frame is dropped in case that both length and CRC fields are not valid or if the destination address does not correspond to the receiving node. The complete receiving algorithm can be seen in Fig. 2.6.b.

In 5G NR fronthaul, high speed Ethernet connected via optical fiber links brings the possibility to carry eCPRI traffic containing IQ user data samples as well as its related

real-time information. The process of transmitting fronthaul traffic through Ethernet follows the same procedures that are typically found in regular Ethernet transmissions. However, the data that is transported from the RRH to the DU is not simply placed inside the data field of the Ethernet frame. Instead, the Ethernet payload field holds fronthaul data wrapped according to the Open-RAN standard and encapsulated inside a protocol called eCPRI. The following section is dedicated to the introduction and study of the eCPRI protocol and how it's used, alongside Ethernet, for fronthaul transport.

2.3 eCPRI

The Common Public Radio Interface (CPRI) is a radio interface which was developed conjointly by leading telecommunication companies such as Nokia, Ericsson, Huawei and NEC. CPRI seeks to define a public specification to transport RF data between the radio equipment (RE), found in the RRH, and the radio equipment control (REC), found in the base station. Furthermore, CPRI provides a physical connection that enables control signaling, synchronization and I/Q sample transport in both electrical and optical interfaces as well as other topologies such as star, ring, daisy-chain and point-to-point. LTE used CPRI based fronthaul transport that supports data rates up to 24Gbps per cell, offering a fronthaul transport latency up to 200 μs, low jitter, tight synchronization and high reliability [5],[27].

CPRI has some down sides that are problematic for 5G networks such as being an expensive technology to deploy, consuming large bandwidth and being statically configured. It also has very strict delay and jitter requirements, making the design process more cumbersome and difficult [7]. 5G traffic demands a flexible and dynamic network and while CPRI fronthaul can be used at different transmission rates, it can not be configurable dynamically. Moreover, it has proved to be inefficient in varying traffic conditions [28]. A possible solution, as presented in [7], is to take advantage of already existing Ethernet interfaces for fronthaul and encapsulate CPRI over Ethernet frames (CoE) even if there is still scrutiny in whether such requirements are met.

In order to address the new transport challenges among RRUs, DU and CU that are introduced in 5G systems and which can not be properly handled by CPRI, the CPRI forum has developed a new specification for fronthaul interfaces called enhanced CPRI (eCPRI). The newly introduced specification enables increased efficiency for 5G mobile networks and it supports more flexibility in the positioning of the functional split inside the physical layer of the base station [12]. The idea with eCPRI is to allow efficient and flexible radio data transmission via a packet based transport fronthaul networks such as Ethernet. eCPRI uses a protocol layer that has room for different User Plane data specific services to the upper layers of the protocol stack. As in CPRI, there are two hardware/software components: eCPRI Radio Equipment Control (eREC) and the eCPRI Radio Equipment (eRE) [17]. Both components can be physically separated, for example the eRE may be closer to the antenna while the eREC can be located in a different site. Compared to CPRI,

eCPRI decreases the data rate requirements between eRE and eREC by decomposing and limiting the complexity of the eRE [12]. This split allows the eREC to process the higher part of the PHY layer and higher layer functions while the eRE holds the lower part of the PHY layer as well as the RF processing.

2.3.1 eCPRI Functional Split

As presented in Sec. 2.1.1, 5G brings a functional split in the RAN that divides its functionality over different network components. There are several options that can be considered depending on the network requirements, as it can be seen in Fig. 2.2. The eCPRI standard also defines five inter-layer functional splits, seen in Fig. 2.7, and are labeled from A to E. Furthermore, there is a set of internal functional splits (intra-PHY) identified as I_D, II_D and I_U are also defined in the standard. The eNB/gNB consist of two units, namely the eREC and the eRE. Some of the splits are not possible to deploy if only two nodes are being used. According to the eCPRI standard [12], features such as carrier aggregation, network MIMO, downlink COMP, uplink L1 Comp joint processing can be efficiently supported by implementing the intra-PHY split. The functional split used in this case corresponds to option E in the figure, independently of which intra-PHY split is used. All the five inter-layer splits introduced in the eCPRI standard are consistent with the functional splits defined in the NR standard and discussed in Sec. 2.1.

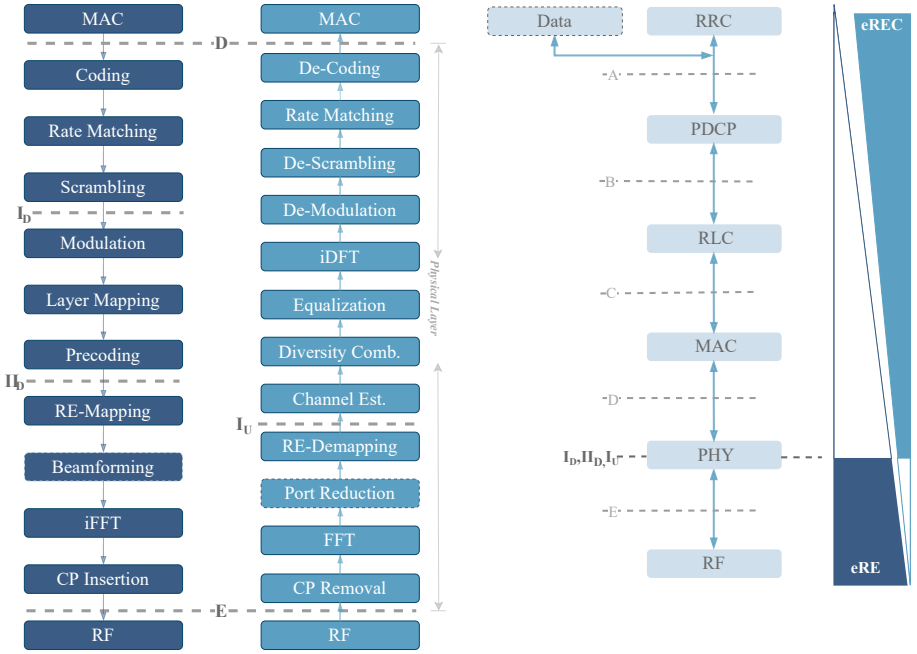


Figure 2.7. *eCPRI splitting options*

The eCPRI intra-PHY split establishes two splits in the downlink direction and one split in the uplink direction and any combination of splits is possible. Fig. 2.7 shows the possible I_D, II_D and I_U splits with the respective processing layers which are included on each division. The figure corresponds only for the data plane processing, however, there is a parallel user data real time control flow which must be also implemented. This parallel flow carries the real-time control data such as modulation scheme, transmission power, beamforming information and anything that is needed for the receiving node to process the user information. The bit rate of the real-time control data depends on which split is implemented and the closer the split is to the MAC layer the more control data needs to be sent between the nodes. The biggest difference between split I_D and split II_D is that the data in split I_D is bit orientated whereas the data in split $II_D$and I_U is IQ orientated. According to [12], one of the major objectives for the new split compared to the classical CPRI functional split, is to lower the bit rates on the fronthaul interface by moving the split upwards, in Fig. 2.7, and conversely, the bit rate for the user plane real-time control data will increase when moving the split towards the MAC layer. The four processes that increase the bit rate are: modulation, beamforming, inverse fast Fourier transform and cyclic prefix process. Moreover, the fronthaul eCPRI traffic is independent on the user traffic profile when lower functional split layers are used [10].

2.3.2 Common Header for User Plane

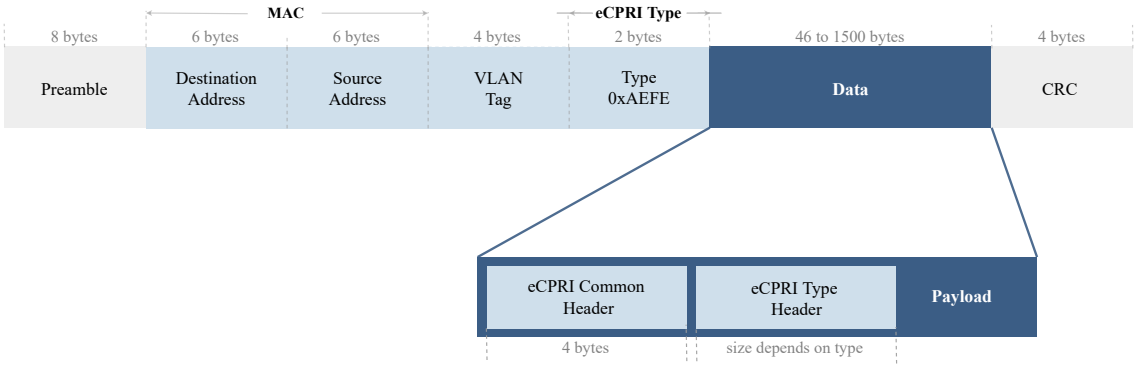
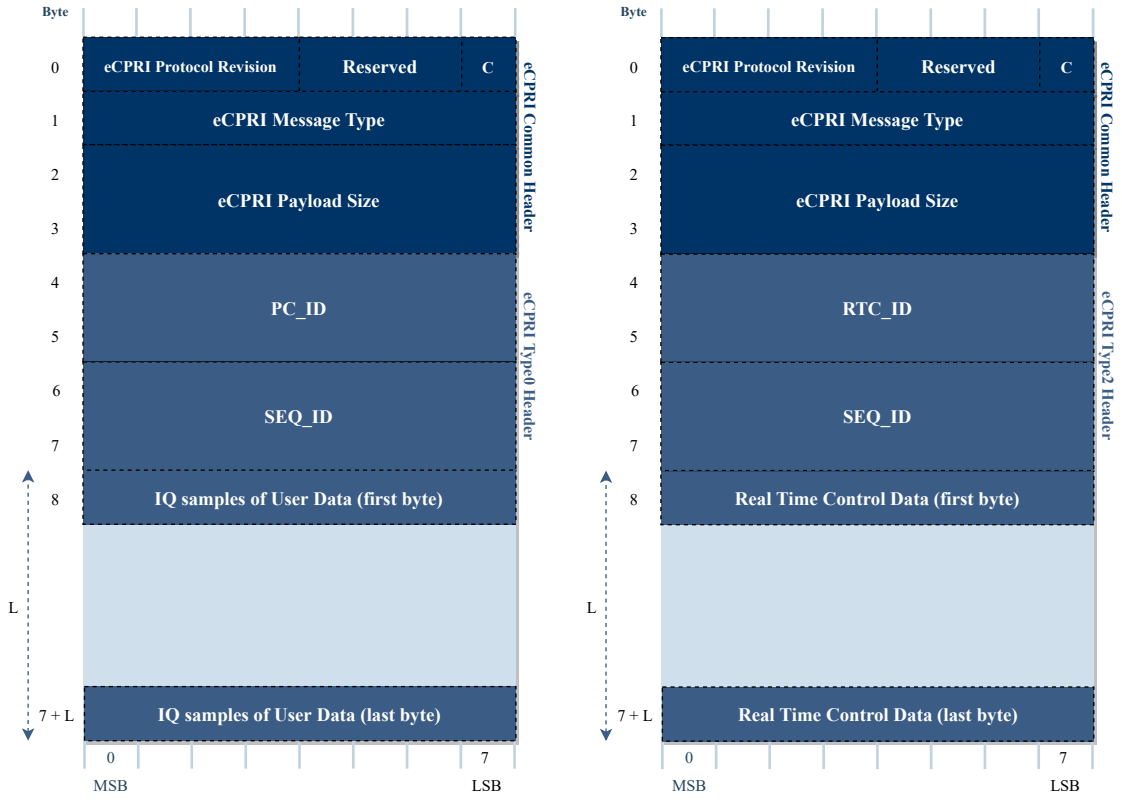


Figure 2.8. *Ethernet frame with eCPRI headers and eCPRI Ethertype*

There are three different information planes found on eCPRI: User plane, C&M (control and management) plane and synchronization plane. The user plane carries information between two nodes using the protocol defined by eCPRI and it does so by using a packed based fronthaul transport such as Ethernet. There are three different data flows which can be carried in the user plane: user data, real-time control data and others. The user data contains user information that is transmitted to/from the base station to the UE. The real-time control data transports time critical management and control information related to the transmitted user data. Others refer to different eCPRI services such as remote reset, user-plane support, etc. The C&M plane transports data related to the management and control of the data flow for the administration, operation and maintenance of the eCPRI

nodes. The control and management information transport occurs within the eREC and the eRE and the information is not time critical. The synchronization plane allows for data flows to synchronize and time information between the different eCPRI nodes for the frame and time alignment.

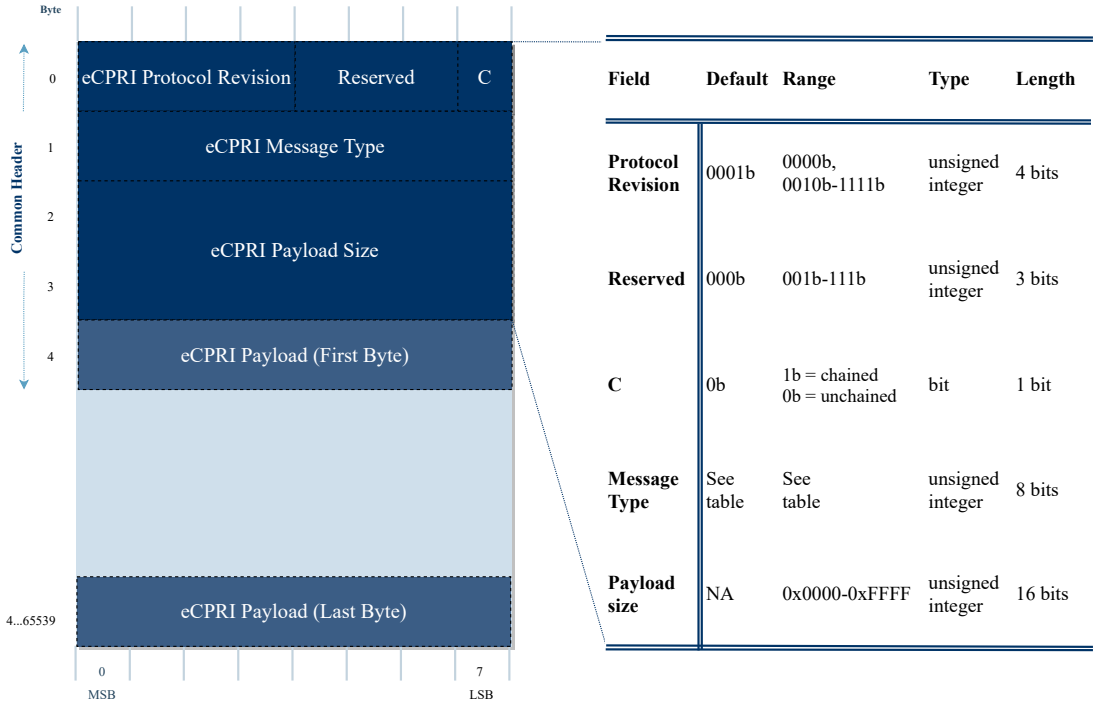
The transfer of user plane information defined by the eCPRI protocol is done using a packet based fronthaul transport network such as Ethernet or IP. According to the eCPRI specification [12], the protocols for C&M and synchronization transport are referenced as proposals. That is, there is room for changes to take place. This thesis focuses primarily on Ethernet transport, which in the context of eCPRI is processed as follows. The eCPRI messages are transmitted using standard Ethernet frames. Each frame consists of transport network layer header with the specific eCPRI Ethertype (0XAEFE) followed by the transport network layer payload which is where the eCPRI common header and type header are embedded. The transport header must have at least one MAC address that is unique, either globally or locally, and all pertinent source and frame priority information to ensure the correct delivery in the network. The length and structure of the eCPRI type header found in the payload layer varies depending on the payload being carried on the eCPRI payload. The eCPRI data field can be padded with zeros to meet the minimum Ethernet frame size requirement of 46 octets if needed [12]. The Ethernet frame containing the eCPRI fields can be seen in Fig. 2.8.



(a) IQ User Data *(b) Real-Time Control Data*

Figure 2.9. *Example of two eCPRI frames with their common and type headers for IQ and real-time control data transport.*

The basic eCPRI encapsulation is made up of two fields called eCPRI common header and eCPRI type header. The common header provides general eCPRI transport information and is always used regardless of the type of data being carried. The eCPRI type header carries important parameters related to the transported data such as different types of identification (specific stream, channel, carrier,etc.), beamforming weight coefficients, numerology structures (symbol, PRBs per symbol, slots, etc), etc. Fig. 2.9 shows an example of two frames containing the eCPRI common header and two different type headers: IQ user data (Fig. 2.9a) and real-time control data (Fig. 2.9b). The transfer sequence for these frames is case specific, although an example case was presented earlier and can be seen in Fig. 1.2 where the frame carrying real-time control data was transmitted first and then the IQ user data carried over several frames was sent after. The transaction is done that order so that the receiving node (RRH in this case) is configured properly to parse and handle the incoming user data. Both eCPRI types are encapsulated in an Ethernet frame such as the one in Fig. 2.8. The transmission of data occurs in "big-endian" byte order, "little-endian" bit order, which means the most significant byte is transmitted first then the byte before the most significant byte and so on until the least significant byte. For example, if one of the fields is 0xEF0ACD01, then the first byte to be transmitted is 0xEF, followed by 0x0A, and so on until 0x01 is transmitted.



Field	Default	Range	Type	Length
Protocol Revision	0001b	0000b, 0010b-1111b	unsigned integer	4 bits
Reserved	000b	001b-111b	unsigned integer	3 bits
C	0b	1b = chained 0b = unchained	bit	1 bit
Message Type	See table	See table	unsigned integer	8 bits
Payload size	NA	0x0000-0xFFFF	unsigned integer	16 bits

Figure 2.10. *eCPRI Common Header with its different field options*

There are 5 fields included in the common header: Protocol Revision, reserved, message concatenation indicator(C), eCPRI message type and eCPRI payload size. The eCPRI protocol revision indicates the eCPRI protocol being used. Its default value -which can be seen in Fig. 2.10 - is for eCPRI version 1.0. This field is only incremented when changes in new releases are not backward compatible with older released versions. This compatibility

mechanism allows the processing unit to have a simple and fast start-up procedure. The second field is reserved and is used as a placeholder for future releases to have the ability to include enhancements or introduce features. It can also be used for vendor specific values if needed. The third field found on the first byte of the frame, called message concatenation indicator, is used to indicate if concatenation of multiple eCPRI messages over a single Ethernet frame is in use. If set to zero, the field indicates that the eCPRI message is the last message inside the eCPRI PDU. Else (set to one), it indicates that there is another message following in the same eCPRI PDU and 0 to 3 padding bytes must be added between messages so that they are 4-byte aligned. The second byte of the frame holds the eCPRI message type that describes the type of data transported in the frame. Table. 2.2 shows all possible data types that can be carried and their brief description. These fields tell the processing unit what type of data is being transported so that the receiving unit knows what data to expect. The different specific parameters, specially those related to fronthaul transport, will be discussed in the next section. The following two bytes in the eCPRI common header, bytes 2 and 3, hold the eCPRI payload size which is the amount of bytes of the eCPRI message without padding. According to [12], the maximum payload size is 2^{16}-1 although the real size is constrained by the transport network. The eCPRI payload field carries the eCPRI type header and is not part of the common header. The length of the payload and fields of the frame depends directly on the type of message that is carried and varies from case to case, specially if concatenation is used.

2.3.3 Message Types for User Plane

Since 5G brings different forms of data that require specific handling, such as different antenna ports (in the case of MIMO) or different numbers of PRBs that change dynamically depending on their SCS (as in the case of mixed-numerology), the transport mechanism must be adaptable and flexible. eCPRI supports different types of data that can be transferred as messages over a communication link, which is perhaps one of the most attractive factors of eCPRI as a solution for fronthaul communications: its capability to transport different types of data over the same medium while being dynamically configured. As mention earlier in the section, the eCPRI type header provides specific information about the type of data being transported so that the processing components can parse and route the information properly. Table 2.2 has a list of the currently supported eCPRI message types according to the eCPRI specification [12].

The table shows the message type number identifier, its name and a brief description of the type of data/service that it carries. There are different options for different transport scenarios, for example a delay measurement estimate (type 5) can be used to measure the latency of two network components. It is done by capturing and sending a time stamp (with a compensation factor) to another node that will capture and send back the time stamp when the signal was received. The initial node can calculate the time difference plus the compensation factor. This can be an useful feature to test latency in network

where time critical applications are needed, as it was discussed in Sec. 2.1. Nonetheless, this thesis focuses on IQ (type0) and real-time control data (type2) transport, so special attention will be given to both cases in the coming sections.

Table 2.2. *eCPRI Message Types*

Type	Name	Description of Usage
0	IQ Data	Transfers time domain or frequency domain IQ samples
1	Bit Sequence	Transfers user data in bit sequence form
2	Real-Time Control Data	Transfers Real-Time control messages
3	Generic Data Transfer	Transfers user plane data or related control between eCPRI nodes
4	Remote Memory Access	Reads and writes from/to specific memory address on a different eCPRI node
5	Delay Measurements	Estimates one-way delay between two eCPRI ports in one direction
6	Remote Reset	Requests the reset of another eCPRI node
7	Event Indication	Indicates on a different node that an event has occurred
8	IWF Start-Up	Starts up the transfer sequence between CPRI nodes
9	IWF Operation	Transfers basic CPRI frames between CPRI nodes
10	IWF Mapping	Negotiates mapping configuration between two IWFs
11	IWF Delay Control	Reports delay values from a remote device
12 - 63	Reserved	Reserved for future eCPRI specifications
64 - 255	Vendor Specific	Reserved to be defined by vendor

Message type0 are specified when IQ samples are transported between the eREC and the eRE in either time or frequency domain. An example of a message type0 and its structure is shown in Fig.2.9a. The first field found in a type0 header is called PC_ID and is used as an identifier for a series of transfer messages containing IQ data. It identifies the data streams from different specific sources such as: an antenna port, a user, a physical channel, etc. The parameter following the PC_ID is called SEQ_ID and is used as an identifier for each message that belongs to a larger series of messages containing IQ data, such as OFDM blocks, sub-carrier blocks, PRBs, etc. For example, if an OFDM symbol is split into 200 frames, the the SEQ_ID field is used as a counter so that the receiving node knows that when SEQ_ID of the incoming frame is equal to 199 (if counting starts from zero) then the last message of that OFDM symbol has been received. The way that PC_ID and SEQ_ID are defined is vendor specific depending on the application and transport that the system is employing. These two parameters will be revisited in Sec. 2.4 and Sec. 3

and will be defined for the scope of this thesis. The third field found in the header is called IQ Samples of User Data and it carries the user In-Phase (I) and quadrature(Q) data pairs in frequency or time domain, depending on the functional split between the eCPRI nodes. All format types of the IQ data that are carried in the frame (i.e bit width, compression definition, number of pair samples, fixed point, floating point, block floating point and others) are vendor specific. Furthermore, all the network nodes that will share information with each other must be configured using the same vendor specific format so that the transmitting and receiving eCPRI nodes can decode the packages using the same parameters.

Fig. 2.9b shows the structure used of an eCPRI type header carrying real-time control data messages, represented as type2. These types of messages are used to transfer real-time control messages that include information for configuration, measurement and control of user data messages. These messages are usually sent before the transmission of type0 messages to inform the remote node on how to process the upcoming type0 data but specific transmission sequences can vary depending on the specific scenario. Type2 messages depend directly on the RAN's functional split, discussed in Sec. 2.1.1, and on the eCPRI functional split shown in Fig.2.7. The first field in the header is called RTC_ID and is used as an identifier of specific configuration, status, measurement & request, indication types, response and control in a series of streams of real-time control messages. The second field found in the type 2 header is a SEQ_ID field. Very much like in the case of type0 messages, the SEQ_ID field in type 2 messages is used as an identifier of each message in a series of real time control data messages such as in links between request and response messages,etc. The specific implementation of the RTC_ID, SEQ_ID and real-time control data fields for type 2 messages are, as in type0 headers, vendor specific.

As discussed over the span of this section, eCPRI is made up of a common header and a type header that are placed inside an Ethernet frame. The type header tells the receiving node what type of data is being transported. However, eCPRI does not provide a specific methodology for arranging the transported payload. For that, Open-RAN has specified a protocol to organize IQ and control payloads -with all their necessary identification types such as compression, bit length, numerology arrangements, etc.- inside an eCPRI layer that can be transported over Ethernet.

2.4 oRAN

The 5G RAN has challenging architectural requirements, such as low layer functional splitting options that require more bandwidth and lower latency, that impose tight constraints in the fronthaul transport. Using software to control mobile and optical segments of the fronthaul has been a proposed solution to such constraints since it allows user and control plane control protocols to work with each other in a simpler and cost effective way [24]. Furthermore, splitting the RAN into different segments depending on the functionality opens the possibility of making the RAN more efficient, smarter and significantly increase its versatility. The proposed architecture to achieve both intents is called O-RAN (open RAN) and its aim is to develop a multi-vendor fronthaul that can be open, provide RAN virtualization, AI enabled RAN access and provide readily available off-the-shelf hardware with minimal proprietary IPs among other services. Moreover, oRAN allows a software oriented infrastructure that enables the network to adapt depending to the QoS requirement of the application and it facilitates the support of IoT networks as well as modern high speed applications in a stand alone and flexible network by being a disaggregated, virtualized, self-driven, application and software oriented network [29].

oRAN introduces agility to the RAN by allowing it to adapt to new applications and scenarios without the need to alter or introduce new hardware components. This sort of ability can eventually allow DevOps technologies into RAN deployment which can enable faster testing and development [25]. oRAN dis-aggregates some functions that have traditionally been in a single node inside the RAN. For example, by using server platforms to implement control plane functions while the real-time processing is accelerated by highly specialized hardware [29]. A key feature of the proposed architecture is to extend software defined networks (SDN) by implementing embedded intelligence through the use of a hierarchical (non-real-time and near-real-time) RAN intelligent controller (RIC) with A1 and E2 interfaces that can execute advanced control functionalities that can leverage analytics and data-driven approaches such as machine learning (ML) and artificial intelligence (AI). The oRAN architecture is built on several decoupled RAN components which were defined by 3GPP, such as $F_1, W_1, E_1.X_2$ and X_1. Some of the new interfaces specified by the O-RAN Alliance are: open fronthaul interface between the DU and RRU, E_2 interface, A_1 interfaces that contain a non-real-time RIC function -called RIC non-RT- and an eNB/gNB that has a near-real-time RIC function -called RIC near-RT.

The eNB/gNB architecture in the oRAN specification introduces new naming conventions for the distributed unit (DU), control unit (CU) and the radio unit (RU): O-DU, O-CU and O-RU. Their distribution is analogue to that which is found in Fig. 2.1. The real-time processing of the user and control plane of the O-RUs are controlled by the O-DU. The O-DU and O-RU are logical nodes that perform functions according to the functional split proposed by oRAN, called split 7-2x, and are mutually connected using a new interface called Lower Layer Split (LLS), which is made up of two sub-layers: the Lower Layer Split Control-Plane (LLS-C) and the Lower Layer Split User-Plane (LLS-U).

2.4.1 ORAN Functional Split

As seen on previous sections, 5G introduces options for a functional split in the RAN (Sec. 2.1.1) and the eCPRI protocol specifies possible split options as well (Sec. 2.3.1). The oRAN architecture selected a single split point, called Split 7-2x, that agrees for the most part with previously discussed RAN splits, except for some PHY functionality in the radio unit that is introduced by oRAN and is not included in the 3GPP study [4]. This split point allows for the pre-coding function to be located either in the O-DU interface, which is assigned the name of Category A, or in the O-RU , which is called Category B. The suggested split can be seen in Fig. 2.11, where the blocks marked with dashed lines are optional depending on the desired category and if it supports beamforming or not. Category A does not support pre-coding in the radio unit whereas category B supports pre-coding in the radio unit and modulation compression. Digital and analog beamforming are options present in both categories. Extensive studies are conducted to improve energy consumption in RAN architectures and traffic for fronthaul fiber-optic transmissions. [14] [20] [18].

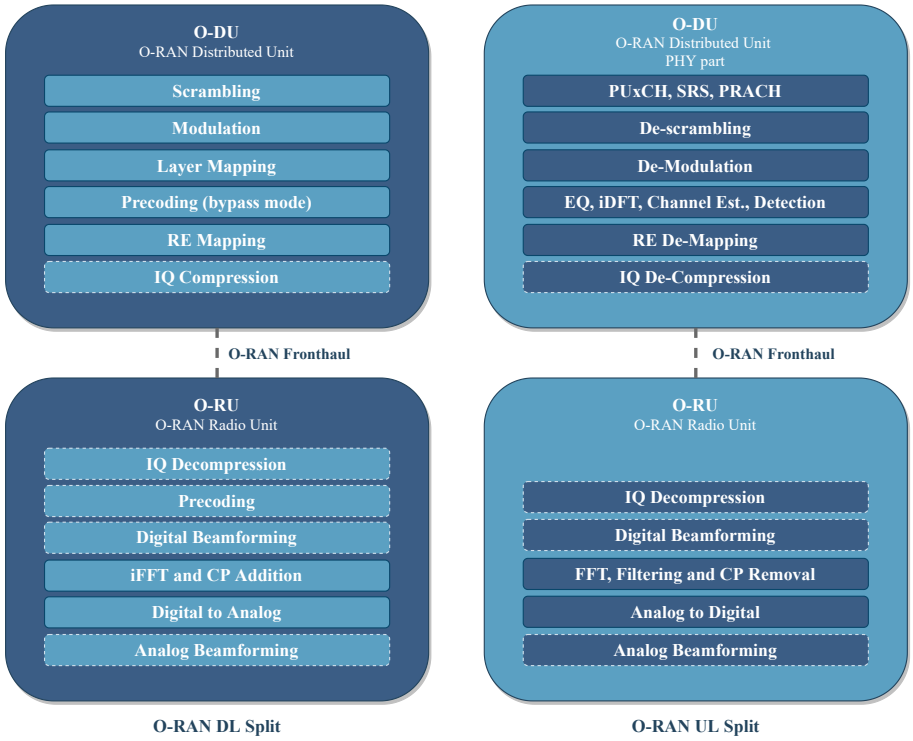


Figure 2.11. *O-RAN Split 7-2x for NR in DL and UL*

The categories are defined as following. For downlink transmission, the O-RU performs ordering of digital beamforming, iFFT, CP addition and phase compensation. If category B is used the O-RU includes pre-coding as well. The remaining of physical layer functions such as resource element mapping, pre-coding, layer mapping, modulation, scrambling,rate matching and coding will reside inside the O-DU. In the case of category A, the pre-coding must be inside the O-DU and the number of pre-coder output spatial streams is 8 or less,

otherwise category B must be used. In case of uplink transmission, seen on the right side of Fig. 2.11, the OFDM phase compensation, FFT, CP removal and digital beamforming are inside the O-RU while the remaining PHY functions, such as resource element de-mapping, equalization, de-modulation, de-scrambling, rate de-matching and de-coding, are placed inside the O-DU.

The split offers benefits for both uplink and downlink such as [26]: interface simplicity, the user plane data is transferred based on resource elements or physical resource blocks which simplifies data mapping;transport bandwidth scalability, 7.2x interface scales based on streams which allows using high number of antennas without higher transport bandwidth; beamforming support, same design supports digital, analog and hybrid beamforming as well as different types of algorithms; Inter-operability, less user specific parameters are used; Advanced receivers and inter-cell coordination are implemented in the O-DU which is simpler; Lower O-RU complexity, since less functions occur at O-RU less real time calculations and memory are required; Future proof-ness, allows new features to be introduced via software upgrades without the need to adjust the hardware; interface and function symmetry, if the same interface and split are used for both downlink and uplink transmission the specification effort can be reduced.

2.4.2 ORAN Encapsulation

U-plane and C-plane messages can be encapsulated over different transport mechanisms such as Ethernet and IP/UDP. In the case of Ethernet, the messages are transmitted over standard frames that define the eCPRI Ethertype (0xAEFE) and carry eCPRI headers and payloads. oRAN stipulates how the data should be arranged inside the payload section of in the eCPRI layer. As presented in Sec. 2.3, the eCPRI common header provides data routing capabilities that indicate the flow type, the sending and receiving ports, if there are concatenated of multiple messages in a single Ethernet packet and the sequence number. However, when the eCPRI protocol was introduced, some definitions were not specified since they are vendor specific. The O-RAN protocol [26] provides a proper definition of those parameters and expands on how they can be used.The eCPRI real time control data (RTC_ID) and the IQ data transfer message series identifier (PC_ID) are defined as component_eAxC identifiers (e_eAxC ID) that identify specific data flows related with each C-Plane (RTC_ID) and U-Plane (PC_ID) messages. The eAxC comes from its CPRI predecessor AxC, which was an antenna carrier, but now an "e" is aggregated to the name that signify "extended" since it can accommodate multiple bands and multiple component carriers. The eAxC is divided into several c_eAxC (component eAxC) parts since multiple O-DU processor can interact with a single eAxC. A single c_eAxC can be used to identify channels that support mixed-numerology - discussed in Sec. 3.3.2- in which case the PRACH can share the same c_eAxC Id or if the channel does not support mixed numerology, then different c_eAxC Ids would be used for each channel that contains a different numerology. The Id value for c_eAxC must be unique among all endpoints of

O-RU of the same direction (transmitting or receiving) even if virtual fronthaul interfaces (i.e VLANs) and multiple physical interfaces are present.

Each eAxC Id is made up of the following Id sub-fields: a sector and band identifier (BandSector_ID), is used to aggregate cell identifiers; a DU_Port_Id, used to differentiate processing units such as baseband cards at the O_DU; a component-carrier identifier (CC_ID), it distinguishes the carrier components supported by the O-RU and a spatial stream identifier (RU_Port_ID) that provides identifiers for spatial streams, mixed numerology, PRACH, special antenna assignments(SRS). Several c_eAxC Ids can share the same BandSector_ID, CC_ID and RU_Port_ID but shall have different values of DU_Port_ID. This parameter sub-fields are specified for eCPRI message type2 (C-Plane, Fig. 2.9b) and type0 (U-Plane,Fig. 2.9a) since the eAxC contains only one spatial stream at a time [26]. They are set via M-Plane messages and their bit-fields have variable length to provide flexibility. The bit field structure can be seen on Table. 2.3, where the length of each parameters is not specified but the order on which they should included is.

Table 2.3. *PC_ID/RTC_ID (eAxC ID) possible bit allocation*

msb															lsb
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DU_Port_Id				BandSector_Id				CC_Id				RU_Port_Id			

The SEQ_ID (sequence Id) is used to provide unique identifiers to order messages. The parameter is 2 bytes long, where the first byte is called sequence Id and the second byte is called sub-sequence id. The sequence Id byte increments with wrapping for every U-plane eAxC DL/UL, C-Plane eAxC DL/UL message and it checks that all the messages are received correctly;it re-organizes them if need be. Table. 2.4 shows the bit-field and bit-length of each of the parameters.

Table 2.4. *Sequence Id (SEQ_ID) bit allocation*

0 (msb)	1	2	3	4	5	6	7 (lsb)
Sequence Id							
E bit	Subsequence Id						

The sub-sequence Id, found in the second byte, consists of a 7 bit sub-sequence fragment counter plus a single bit called E-bit that indicates a last fragment. The sub-sequence Id definition depends on the fragmentation method being used in the transmission. There are currently two methods for fragmentation supported by [26]: Application layer fragmentation and transport layer fragmentation. In Application layer fragmentation, the application layer ensures that the transport messages fit within the necessary transport payload size, Ethernet in this case, the sub-sequence identifier is always set to zero and the E-bit is always set to 1, since every message contains the whole fragment. In transport layer fragmentation, U-plane messages can be longer than the underlying transport method allowance, and thus are needed to fragmented so that one message containing U-data is transported in several frames. For transport layer, the sub-sequence counter,

starting at zero, increments on each frame containing a fragment of a U-Plane message and the E-bit is set to 1 only when the last fragment of the message is sent. The sequence Id in both fragmentation cases is generated by the fronthaul interface in either the O-RU or the O-DU and the counter is unique per each eAxC (PC_ID/RTC_ID). An example of an application layer fragmentation can be seen early on in this document in Fig. 1.2, where a C-plane message and several U-plane messages are concatenated in separate Ethernet frames in fronthaul transmission.

Table 2.5. *C-Plane section types*

Type	Target Scenario	Description of Usage
0	Unused Resource Blocks or symbols in DL or UL	Tells the O-RU that certain RB or symbols will not be used. Transmission may be idle for some period of time
1	DL/UL Radio Channels	Used for channels not requiring time or frequency offsets (non-mixed-numerology)
2	Reserved for Future Use	NA
3	PRACH and Mixed Numerology Channels	Channels that require time of frequency offsets or special SCS values
4	Reserved for Future Use	NA
5	UE Scheduling information (UE ID assignment)	Provides scheduling information for UE-IDs
6	Channel Information	Sends UE-specific channel information from O-DU to the O-RU
7	LAA	Messages between O-DU and O-RU to configure LBT for PDSCH/DRS transmission
8 - 255	Reserved for Future Use	NA

Table. 2.5 lists the different data types and their target scenario. C-plane messages are carried inside either eCPRI or IEEE 1914.3 transport protocols. The messages are encapsulated using a two layer header; the first header is the eCPRI common header, including the fields that stipulate the message type and the second header is an application layer that defines U-plane data characteristics. There are 5 elementary procedures defined for C-plane messages [26]: Scheduling and beamforming transfer procedure, used to transmit data associated control information to process user data between O-DU and O-RU transmissions; mixed numerology and PRACH handling, used when sub-carrier spacing is different; DL pre-coding configurations, used to specify Category B pre-coding operations in the O-RU (becomes irrelevant for Category A since pre-coding is done in the O-DU); LAA commands transfer procedure, used to exchange C-plane messages to support the LAA feature in the O-RU/O-DU; Symbol numbering and duration procedure, used to determine the symbol duration since they are different between NR and LTE. These elementary procedures are carried out by using different section types, inside the Ethernet frames, describing the characteristics of the U-plane data being handled. All the sections are 4-byte aligned and are transmitted in network byte order (big-endian byte, little endian bit order).

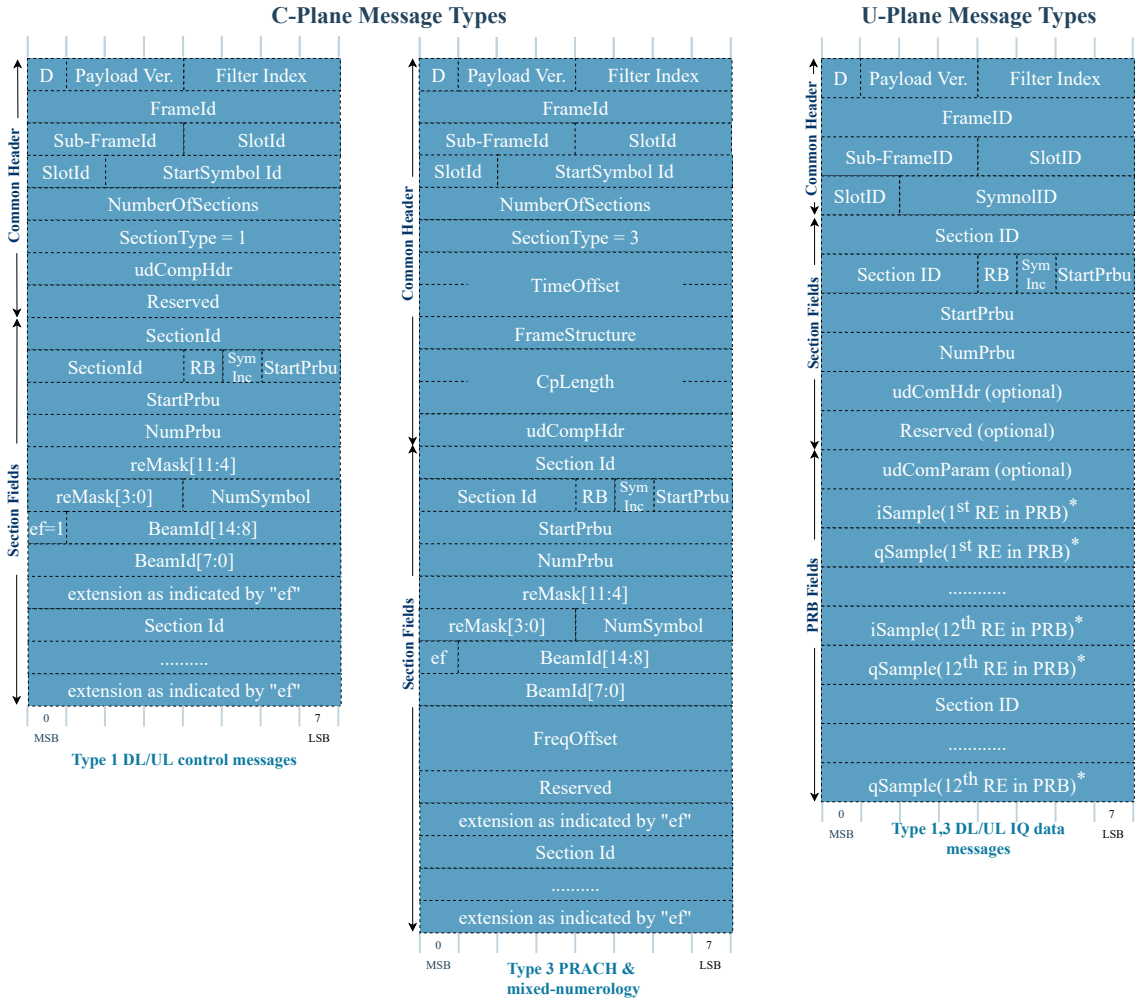


Figure 2.12. *Example of oRAN encapsulated frames carrying c-Plane and U-plane messages*

C-plane messages share a common format that is made up of a transport layer and an application layer. The application layer is inside the transport layer's payload and it consists of a common header followed by a section field that is specific to the section type. In order to save overhead, each frame can contain one application header and multiple concatenated section fields (of the same type). Fig. 2.12 shows an example of type1 and type3 C-plane messages. They share some of the same common header fields but some are different depending on their specified type. The definition of some of the most important fields of the frames are presented as follows; for a detailed description of all the fields as well as their value ranges can be found in the oRAN standard [26]. Starting from the first field on the first byte, D (direction), is used to indicate the gNB data direction. Its set to zero when the data flow is uplink (RX) and set to 1 when the data flow is downlink (TX). The payload version field defines the oRAN protocol version used in that specific flow. The filter index defines an index to the channel filter that will be used for IQ data for both UL and DL. The frame id field, found on the second byte of the frame, contains counts for 10 millisecond frames that wrap every 2.56 seconds. Then, the sub-frame identifier field, found in the first half of the third byte, counts for

1 millisecond sub-frames inside the 10 millisecond frame. The second half of the third byte and the first two bits of the 4th byte carry a slot identifier field that holds the slot number inside a 1 millisecond sub-frame. The current specification allows a maximum of 16 slots and all other values are reserved for future use. The start symbol identifier, which proceeds the slot id field, identifies the first symbol within a slot. The fifth byte holds a field called the number of sections, which tells the amount of data sections inside the C-plane message. Finally, the section type determines the characteristics of the data that is being transferred in the U-plane messages. All the currently supported section types can be seen in Table. 2.5. Furthermore, the frame carrying type3 control data, for PRACH and mixed-numerology, is a good example of the malleability that oRAN brings and that is needed by 5G fronthaul transport since different numerologies, with varying cyclic prefix and beamforming streams, can be parsed dynamically. Moreover, the parsing and configuration of the transport is done using software, therefore, minimal to no hardware components have to be altered in order for the transmission to change parameters. As long as the configuration frames are sent before the frames carrying IQ data, the parsing component will be able to adjust to the new configuration of the data. This scenario was used as an introductory case for this thesis (Fig.1.2), where a C-plane frame was sent first and then 9 U-plane messages followed.

Three parameters that are particularly important for the focus of this thesis since they are unique to section type 3 messages and provide information about the contained numerology are: TimeOffset, FreqOffset and FrameStructure. TimeOffset defines the the time offset from the start of the slot to the start of the cyclic prefix (CP) in number of samples. FreqOffset specifies the offset with respect to the carrier center frenquency before any filtering and FFT processing and is used individually per every data section. FrameStructure indicates the structure of the frame by specifying the FFT/iFFT sized that is used (if used) and the subcarrier spacing. Table. 2.6 provides the bit allocation of how the parameter is arranged inside the frame. The 4 most significant bits represent a bit value that is mapped to an FFT/iFFT size which is specified on Table. 2.7. The 4 least significant bits represent a bit value that is mapped to a subcarrier spacing which is specified on Table. 2.8.

Table 2.6. *Frame Structure Bit Allocation*

0 (msb)	1	2	3	4	5	6	7 (lsb)
FFT size				μ (*subcarrier spacing*)			

U-plane messages are also encapsulated using the same two layer headers, eCPRI and application layer including the necessary fields. The common frame format can be seen in Fig. 2.12. The U-plane messages use some of the same fields for the common header as the C-plane messages. There are two new sections present: The section header fields and the PRB fields. The section header fields carries: the section identifier, to identify individual data sections described within the C-plane message; Resource block (RB), used to indicate if all or every other RBs are used; symbol number increment command (symInc), indicates

Table 2.7. *FFT size*

Bit Representation	FFT/iFFT size
0b0000	no FFT/iFFT processing
0b0001 - 0b0110	Reserved
0b0111	128
0b1000	256
0b1001	512
0b1010	1024
0b1011	2048
0b1100	4096
0b1101	1536
0b1110, 0b1111	Reserved

Table 2.8. *Subcarrier spacing*

Bit Representation	μ	Δf [kHz]	Slots per Subframe	Slot Length
0000	0	15	1	1 ms
0001	1	30	2	500 µs
0010	2	60	4	250 µs
0011	3	120	8	125 µs
0100	4	240	16	62.5 µs
0101 - 1011	NA	Reserved	Reserved	Reserved
1100	NA	1.25	1	1 ms
1101	NA	3.75 (LTE)	1	1 ms
1110	NA	5	1	1 ms
1111	NA	7.5 (LTE)	1	1 ms

what symbol numbers are relevant to a specific section id; Start PRB, to show the starting PRB of user plane data section; Number PRB , to define the number of upcoming PRBs per user plane section; user data compression header (udCompHdr), indicates the compression method and the bit width for the PRB's IQ data which is transported; Lastly, 1 byte of reserved space for future definition. The PRB fields has two sections: udComParam and IQ samples. The user data compression parameter (udCompParam) provides information related to the compression method defined by the comMeth that was specified in the common header. This bit provides, for example, the exponent for a block floating point compression, shift value, etc. The bytes after that store the IQ samples in either time or frequency domain. Furthermore, adding padding bits in the I and Q sample data is possible in order to make it 1 byte aligned. The IQ samples are organized by resource elements and their corresponding PRB. The REs that contain zero values or that are blanked are expected to be transmitted as well, since the data parser that receives the data expect 12 complex resource elements per resource block.

Fig. 2.12 shows a symbolic positioning for the data inside the PRB fields since the bit length of the IQ samples can be different depending on the used compression. The ud-

Table 2.9. *udCompHdr layout*

0 (msb)	1	2	3	4	5	6	7 (lsb)
udIqWidth				udCompMeth			

Table 2.10. *udIqWidth definition*

udIqWidth	Bit Width of each I and each Q
0000 - 1111b	Number describes width in bits of I and Q except for 0 which means 16 bits for each. udIqWidth = 0000b means I and Q are 16 bits wide udIqWidth = 0001b means I and Q are 1 bit wide udIqWidth = 0010b means I and Q are 2 bits wide udIqWidth = 1111b means I and Q are 15 bits wide

Table 2.11. *udCompMeth and udCompParam definitions*

udCompMeth	Compression	udIqWidth bitwidth of each	udCompParam 0 (msb) to 3 : 4 to 7 (lsb)
0000b	none	uncompressed IQ	absent
0001b	block floating point	IQ mantissa value	zeroes : exponent
0010b	block scaling	IQ scale value	sblockScaler
0011b	μ-law	compressed IQ	compBitWidth : compShift
0100b	modulation	compressed IQ	absent
0100b - 1111b	reserved	reserved	zeroes

CompHdr byte, found in the section field, is composed of two equally sized fields, shown in the Table. 2.9, called The idIqWidth and udCompMeth. The idIqWidth describes the width, in bits, of both I and Q components as shown in Table. 2.10 and the udCompMeth determines the type of compression implemented for the I and Q components. Furthermore, the udCompMeth determines what the I and Q bit width in idIqWidth represents, as it can be seen on Table. 2.11 and how to define the udCompParam field. For example, if the udComHdr is defined to be 0x90, it corresponds to an udCompMeth value of 0x0 (0000b) and an idIqWidth value of 0x9 (1001b), therefore, by looking at the tables, there is no data compression defined and each I and Q component are simply 9 bit wide and in this case the udCompParam is not used at all. On the other hand, if the udComHdr has a value of 0x91, block floating point compression is defined where the idIqWidth size specifies a mantissa value of 9 bits for each I and Q components and the udCompParam's four least significant bits define the exponent size for the compression.

All the information provided up to this point is relevant and important to understand in order to properly run the tests and analyse the results. The frame mappings are specially important since they provide details about the waves encapsulated in the O-RAN frames and asses the correctness of the model's output.

3 5G FRONTHAUL MODELLING

This chapter presents the modelling workflow and configuration that was implemented as the main focus of this thesis work. It aims to provide enough details so that the reader understands the mechanics of how the model works and what it aims to achieve. The results are then analysed and discussed.

3.1 Overview of Fronthaul Modelling

Fronthaul modelling is a flexible and dedicated tool to examine interesting processing units in different scopes of a general system. The granularity and focus of the model depends on the complexity of the IP and the scope of the analysis. For example, more complex IP's might require more specific dedicated processing units in the model. The complexity of an IP can be related to the physical demands of the implementation, for example memory sizes for data storage or the complexity the algorithms. Fig. 3.1 presents a general idea of possible sub-system blocks that can be found in a fronthaul scenario. The IQ and configuration packets flow in both UL and DL directions. A link model can also be included where different parameters can be included, for example the effects of distance (in a fiber or coaxial connection), noise, jitter, etc.

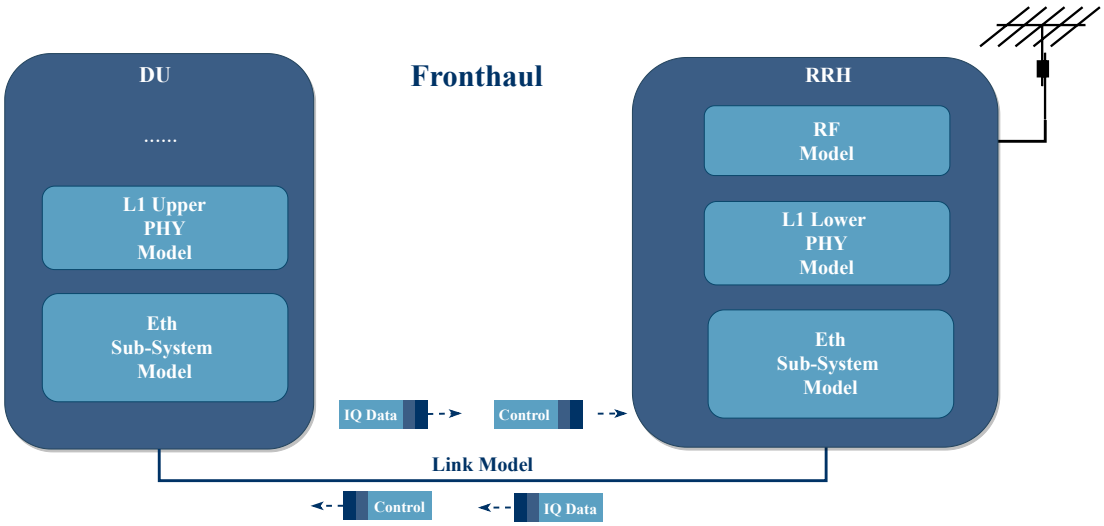


Figure 3.1. *System Level View of different Possible IP Models*

The Ethernet subsystem IP model provides a bridge between data from the DU to the RRH and vice versa. This opens the opportunity for system designers and verification engineers to have an idea of how the data is processed and what output to expect in different locations of the system. For example, the Ethernet IP can work as a bridge between IP models from the L1 upper PHY with the L1 lower PHY models to allow a continuous stream of data which is generated in higher layers and can be analysed in every step until the antenna. A fundamental feature of the model is it's scalability and portability. That is, features can be easily added or removed from the source code to provide flexibility in the processing. Moreover, the processing standards are continuously changing as new emerging technologies or features are implemented in the RAN. As 5G starts to become more mature, there can be the need to include new features or to make corrections in how the processing must be done by the physical implementation. This is greatly alleviated by exploring and implementing changes in the model first in order to understand how the rest of the system will respond to such changes. This is both cost effective and time efficient.

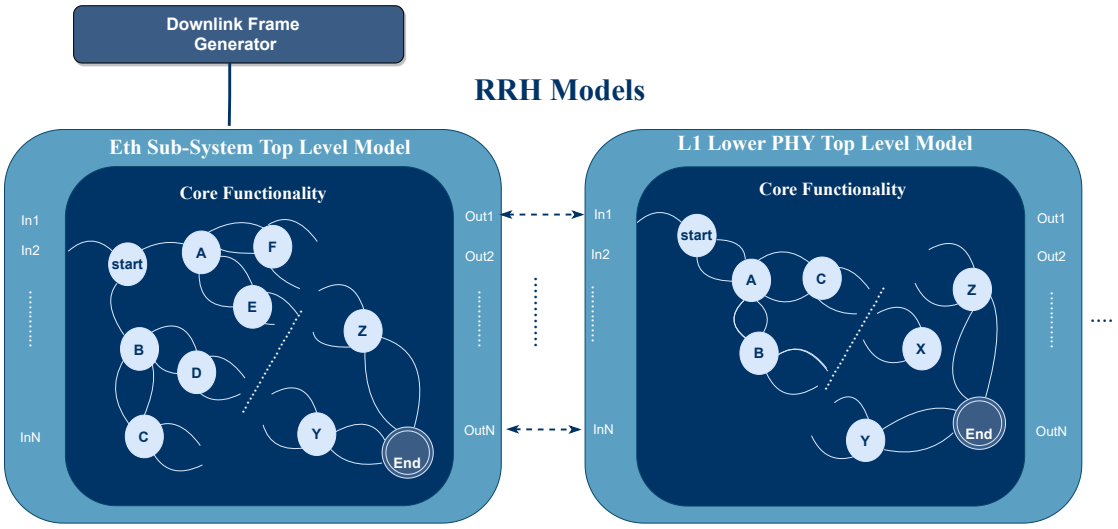


Figure 3.2. *Overview of interaction between Eth and L1 Lower Top Level models*

Fig. 3.2 shows an overview of a simplified fronthaul modelling flow and puts in perspective the location of the Ethernet IP model. In our particular case, no model for the DU was used but instead data generators created the c-plane and u-plane messages that worked as input. The two generators used to test different types of data were Keysight's Open RAN studio/ Signal Studio and Nokia's in house generator. Furthermore, this work assumed perfect linking conditions: no time delays, distortion or noise induced by the connectivity link (as seen on Fig. 3.2) although they can easily be added if need be. Each sub-system block can be programmed to take a variable amount of inputs, in the form of frame types and model configuration files in different formats, and writes an equal amount of inputs to the next processing block. This functionality is scalable and dynamic in principle but it can be hard coded to limit the variability of the model if the user wants to analyse only specific configurations.

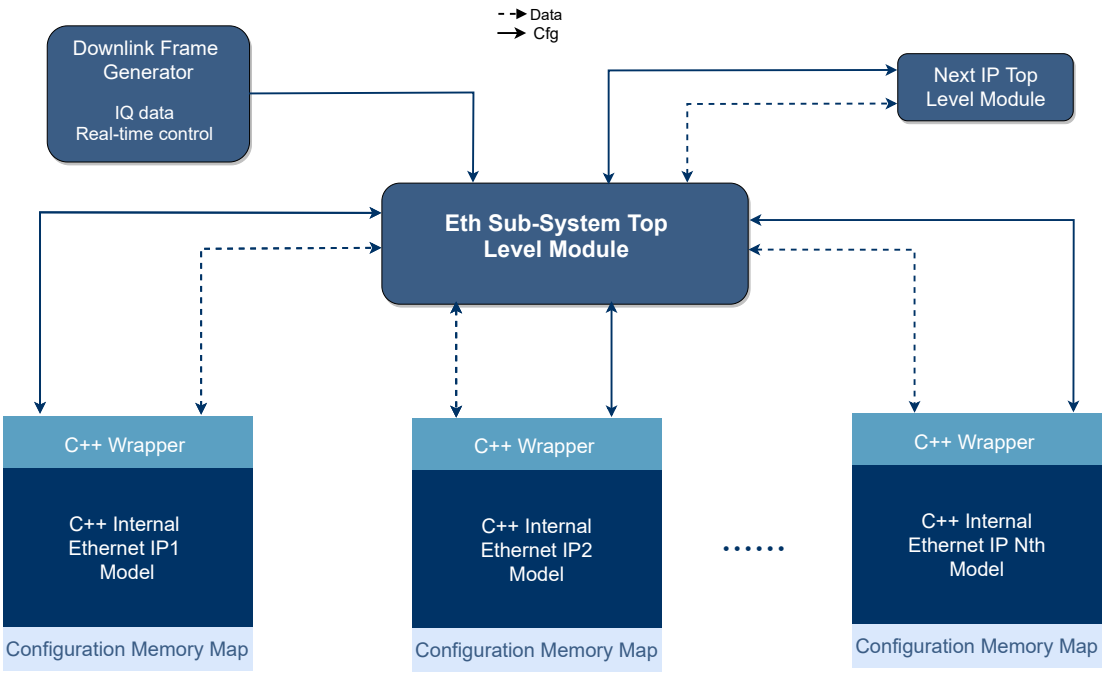


Figure 3.3. *Top Level module interaction with internal models from a sub-system IP*

The modelled IP subsystems can be broken into smaller parts that execute specific processing tasks inside the IP. The are all tied together by a top level module that orchestrates the data and configuration input files within the sub-system IPs, very much so as they are done in the real implementation. Each sub-system IP module has access to the outside environment via wrappers that can be written in a variety of programming and hardware languages, such as Python, C/C++, Ruby, System Verilog, etc. The amount of internal IPs that can be modelled depends on the complexity of the system and the execution demands. Fig. 3.3 shows an example of an Ethernet top level module. The input configuration and data files are grabbed by the top level from the generator(s) and distributed to the pertinent IP models. The output of some models work as input for the upcoming models and so on. In some cases all the IP blocks can share the same configuration memory map or they can use their own, depending on the case desired by the user.

The IP logic is done using C++ and it is compiled as a shared object that can be instantiated by a top level module or by a test bench. This provides the opportunity to connect different models either through software or by generating output text files with the processed data which can be fed into the next processing block. This can provide a very easy way to tweak processing parameters and examine different scenarios. Furthermore, the objects can be instantiated at run time and can even instantiate objects imported from other libraries if need be. For example, systems can be created at run time by deploying a dynamic arrangement of processing units based on the input data and configuration parameters. This feature is very handy particularly at earlier stages of system architectural design or when new features need to be tested in research.

The Ethernet IP model takes pcap files as input and writes the payload and configuration to the next IP block through AXI4 signals. Furthermore, the model is required to be configured with specific processing parameters which are used in the real implementation of the subsystem. The configuration is written in an internal register memory and is accessed at run time in order to properly define the behaviour and processing of the system for the given test-case.

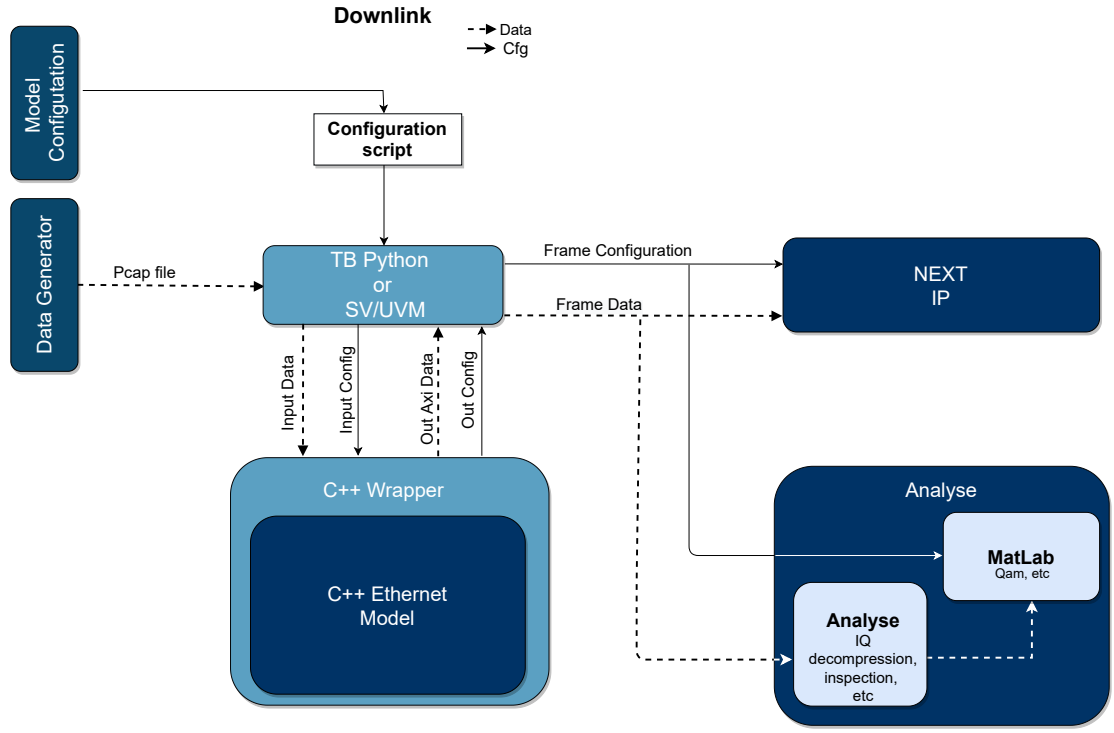


Figure 3.4. *Detailed Modelling Workflow and testing*

In this particular research work, the shared object containing the Ethernet IP model was instantiated by a python test bench which was in charge of configuring, feeding the input data and fetching the output signals. Fig. 3.4 depicts in general the workflow used by the model. The parameter configurations are generated and processed by using a configuration script that is fed into the test bench. The test-bench feeds the configuration definition to the IP model which creates a memory map inside and sets the register values so that they can be accessed at run-time. The data generator takes waves (i.e. OFDM) and creates O-RAN encapsulated pcap files of such waves. Upon input, the C++ model processes the frames, parses the pertinent headers, processes the payload, arranges it in a particular way[1] and then it writes the output via AXI4 signals, which are saved as .txt files in this case. The output files can be fed into other IP models and can also be analysed. The analysis module contains scripts that arrange the data, decompress the written payload and read the configurations in order to plot the written data in Matlab/Octave. The finalized model is rather extensive; it contains over 2500 lines of C++ code and over 1000 lines of python code and it continues to grow as more features are added in.

[1]The specific processing logic of the IP as well as more in-depth details are confidential.

3.2 System Configuration and Input Cases

The system configuration definition is done in a way that matches the required processing for the input data. The selection of frame type, compression header size, number of bytes per message, etc. are dynamically changed at run time. However, there are hardware configurations that must be hard-coded into the model so that it can know what to expect. For example, the memory space that must be allocated for the AXI4 buffers and transactions. The following subsections describe how the input data is configured for single and mixed numerology cases and what type of data will be processed in the model.

3.2.1 Single Numerology Configuration

This research focused in processing single and mixed numerology frames with different resource mapping and channel allocation. This subsection will focus on the generated data for single numerology, as seen in Fig. 3.5. The idea is to generate a simple test case in order to explore the functionality and the performance of the fronthaul model.

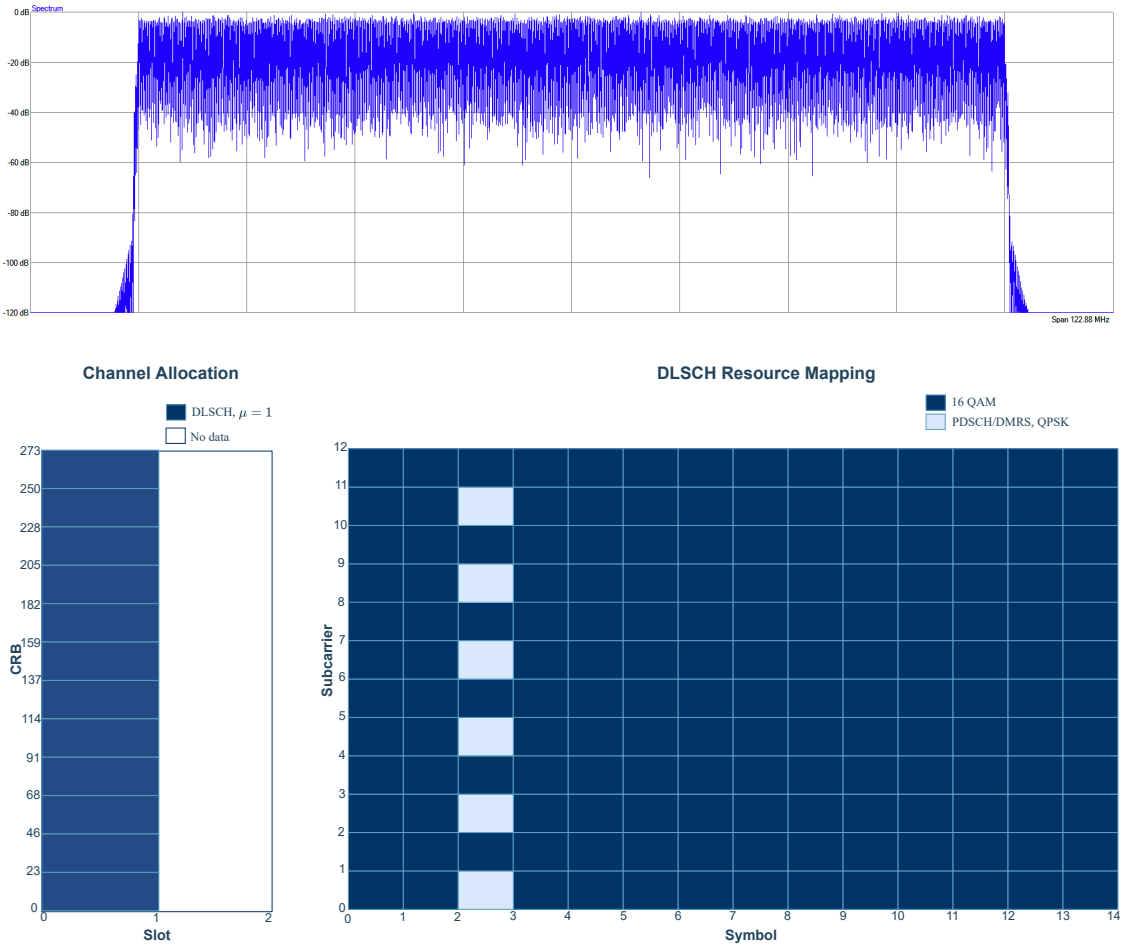


Figure 3.5. *Single Numerology channel allocation for 273 PRBs used as input to the model.*

The first test case, shown in Fig. 3.5, contains 273 PRB's carrying 16-QAM constellations and has an available bandwidth of FR1 100MHz. The subcarrier spacing is 30kHz, which yields to 1 slot per millisecond and each slot contains 14 symbols each with 12 subcarriers. The generator creates 1 pcap file that contains 13 messages that correspond to different PRB ranges within the slot structure.

PDSCH/DMRS signals are also included in the generated data and they are mapped to a QPSK constellation even though its usefulness is outside the scope of this work. This test case is similar to something that could be transmitted and received in LTE. For example, in the case of non-stand alone 5G systems, single numerology processing must be supported by the fronthaul connectivity. The same channel allocation data was encapsulated in bit sizes ranging from 2 bit all the way up to 16 bit, both in block floating point and uncompressed format. This was done to explore the different timing, throughput and theoretical power behavior of the model in regards to different compression types and bit length.

3.2.2 Mixed Numerology Configuration

As seen earlier on in Fig.1.1, mixed numerology is a feature introduced in 5G networks that introduces very sought after communication possibilities, such as tailoring speed and throughput services based on the demands of the user. This sort of dynamic allocation brought challenges in the way fronthaul connectivity is done; links must process data smarter, faster and more efficiently. Back in section 3.3.2 some basic mixed numerology theory was introduced and the same configuration and resource allocation principles are applied when processing mixed numerology frames in this subsection.

For this work, 150 PRBs were generated for mixed numerology testing and not 273 as in the single numerology case. This is simply so that the channel allocation and spectrum waves can be easier to identify for each numerology since they carry the same amount of PRBs. For example, is easy to identify small bandwidth but longer time or bigger bandwidth but shorter time and so on when looking at the resource allocation figure (Fig. 3.6). Furthermore, the numerologies were chosen to carry different QAM constellations such that different features can be more easily exposed when we analyse the received data.

The input data for the mixed numerology example case is shown in Fig. 3.6, where the spectrum of the waves as well as the channel allocation on different numerologies are shown. The bandwidth configuration for the mixed numerology carrier, as in the single numerology case, is also FR1 100MHz and there are 3 BWP (bandwidth parts) configured in this case, all which carry 50 PRB's. BWP1 is allocated to carry a 30kHz SCS which spans to a bandwidth of 98.28MHz, BWP2 is allocated to carry a 15kHz SCS which spans

to a bandwidth of 48.60MHz and BWP3 is allocated to carry a 60kHz (normal CP) which spans to a bandwidth of 97.20MHz.

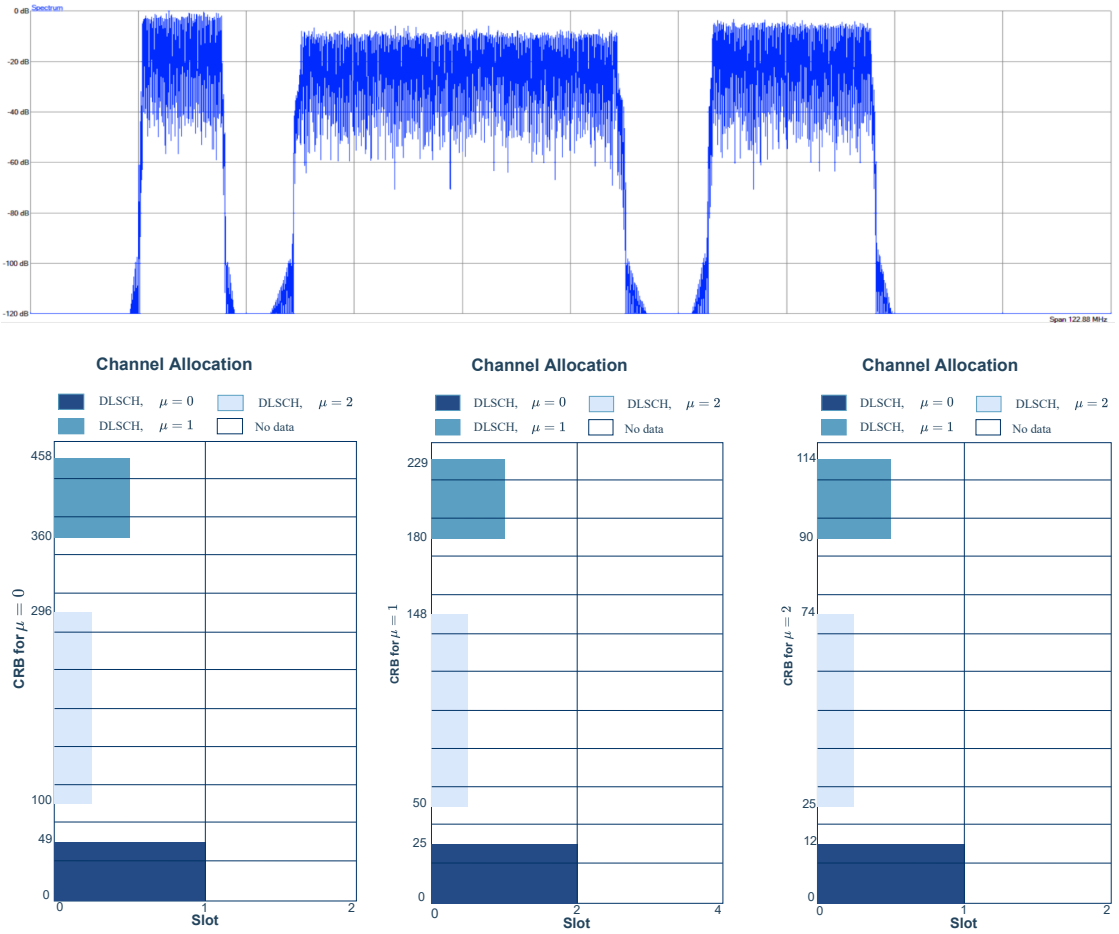


Figure 3.6. *Mixed Numerology channel allocation for 150 PRBs used as input to the model.*

The spectrum seen on Fig. 3.6 clearly shows the three BWP's with the different numerologies. The left most wave carries the lower SCS, 15kHz, the center wave corresponds to 60kHz and the rightmost wave corresponds to 30kHz. The channel allocation varies depending on the numerology that is being used and agrees to the introduction shown in Sec. 3.3.2 about the different types of numerologies. Table. 2.1 and Fig. 2.3 show how different numerologies map to different slots and subcarrier spacings.

The three figures that show the channel allocation, in Fig. 3.6, show how the CRB spacing maps differently depending on the numerology. For example, in the case of μ = 0, 50 CRB slots are taken for the frequency slot that corresponds to 15kHz. In the case of μ = 1 the same 50 slots now map to 25 CRB slots and 12 CRB correspond for the μ = 2 case. The QAM constellations mapped to the different numerologies are: 16-QAM for μ = 0, 64-QAM for μ = 1 and 256-QAM for μ = 2. The mixed numerology payload was also 9-bit block floating point compressed.

3.3 Results and Discussion

The results from the simulations are divided into single and mixed numerology and are presented as follows for each section: The type2 c-plane data is presented in a table, the type0 u-plane messages are written in a table and plotted and the data type performance is analysed for different compression methods and bit lengths. Then, the model execution time and model throughput are studied and discussed. Finally, the digital power scaling is presented and analysed for different bit lengths. The IQ data is shown in three forms: its raw form after decompression, IQ samples provided by Keysight's Open RAN studio generator (that work as input) and IQ samples obtained from a Matlab normalization.

The Matlab normalization factor of the raw samples targeted to have a signal peak power of 1 watt[2]. In the single numerology case, plots are presented both for the raw and normalized representations of the data. In the mixed numerology case, only the raw symbols are presented for the different QAM constellations. The aforementioned tables show the payload data in the format as they come out of the analysis module. This is done in order to maintain the analysis as simple and to the point as possible. The raw output files contain the payload data written over AXI4 address signals, strobe signals, etc. and are used to organize the payload inside a hardware memory space.

In both cases, single and mixed numerology, the first three PRBs are taken as an example to read and analyse the points that are transmitted. Then, a plot containing all the available PRBs is presented to show the full constellation. The first three PRBs do not pose any advantage compared to any other PRBs (i.e PRB number 50 or 100) but they were chosen for simplicity. The hardware used for running the simulations was a Lenovo Thinkpad laptop with 8GB of RAM and an Intel i7-7700HQ CPU with 2.80GHz clock speed and 4 processing cores. The operating system was Windows 10.

3.3.1 Single Numerology Results

C-Plane Processed Data

Table 3.1 shows the parameters recovered for the c-plane single numerology message. The rtcId is the same id as the eAxC ID, which in this case is 0x1. This works as an identifier for the antenna port for which the data belongs to. The sequenceId is 0, representing the first message of the sequence. This parameter is used to keep track of the order of the messages. Since we generated only one c-plane message, no more sequenceId's are expected. E is equal to zero, which it means that there is no fragmentation and this message is complete for its first sequence. Frame, Subframe and slot Ids are identifiers and are all set to zero since this is the first transmission.

[2]The normalization factor used by Keysight is confidential.

Table 3.1. *c-plane recovered message for single numerology*

Parameter	Generated	Recovered
Message	1	1
rtcId	0x01	0x01
SequenceID	0	0
E	1	1
subSeqid	0	0
FrameId	0	0
SubframeId	0	0
SlotId	0	0
startSybolid	0	0
numberOfSections	1	1
sectionType	1	1
udCompHdr	0x91	0x91
sectionId	0	0
rb	0	0
symInc	0	0
startPrbc	0	0
numPrbc	0	0
reMask	0b111111111111	0b111111111111
numSymbol	14	14
ef	0b0	0b0
beamId	0	0

The numberOfSections is set to one, which represents the number of sectionTypes contained in this message. The sectionType used is 1 (type1) which agrees with the single numerology payload that is being transmitted. udCompHdr is set to 0x91 which means 9-bit mantissa using block floating point compression (see Tables 2.9, 2.11 and 2.10 for details). StartPrbc indicates in which PRB the payload begins, and numPrbc indicates how many PRBs are encapsulated in the u-plane message. A value of 0x0 in the numPrbc field indicates all PRBs, since it does not have to be split between different u-plane messages.

According to Fig. 3.5, there are 273 PRB's, so we can expect each u-plane message to contain 273 PRBs with data. The parameter reMask is a mask for resource elements whithin each PRB. All the bits set to 0 indicate that the RE is not applicable. In this case they are all applicable since they all contain data. BeamId and ef define extensions and beamforming weights, they are set to 0 since no beamforming is used in this transmission. For more details about all the parameters, please refer back to Sec 2.4.2.

U-Plane Processed Data

Fig. 3.7 shows the constellation plots of the raw recovered data for different PRBs. Fig. 3.7a shows the recovered data for PRB0. As it can be seen, not all the points for a 16-QAM constellation are there since one PRB alone is not enough to carry all the points. However, the graph shows a noticeable resemblance. Fig. 3.7b and Fig. 3.7c show the plot for the recovered points of PRB1 and PRB2, respectively. They are different points than the ones on the previous plot but still agree with the location of symbols corresponding to a 16-QAM constellation. Finally, Fig. 3.7d is a plot for all the 273 recovered PRBs. Here we can see all the points in the full 16-QAM constellation which is encapsulated in all the PRBs of the message.

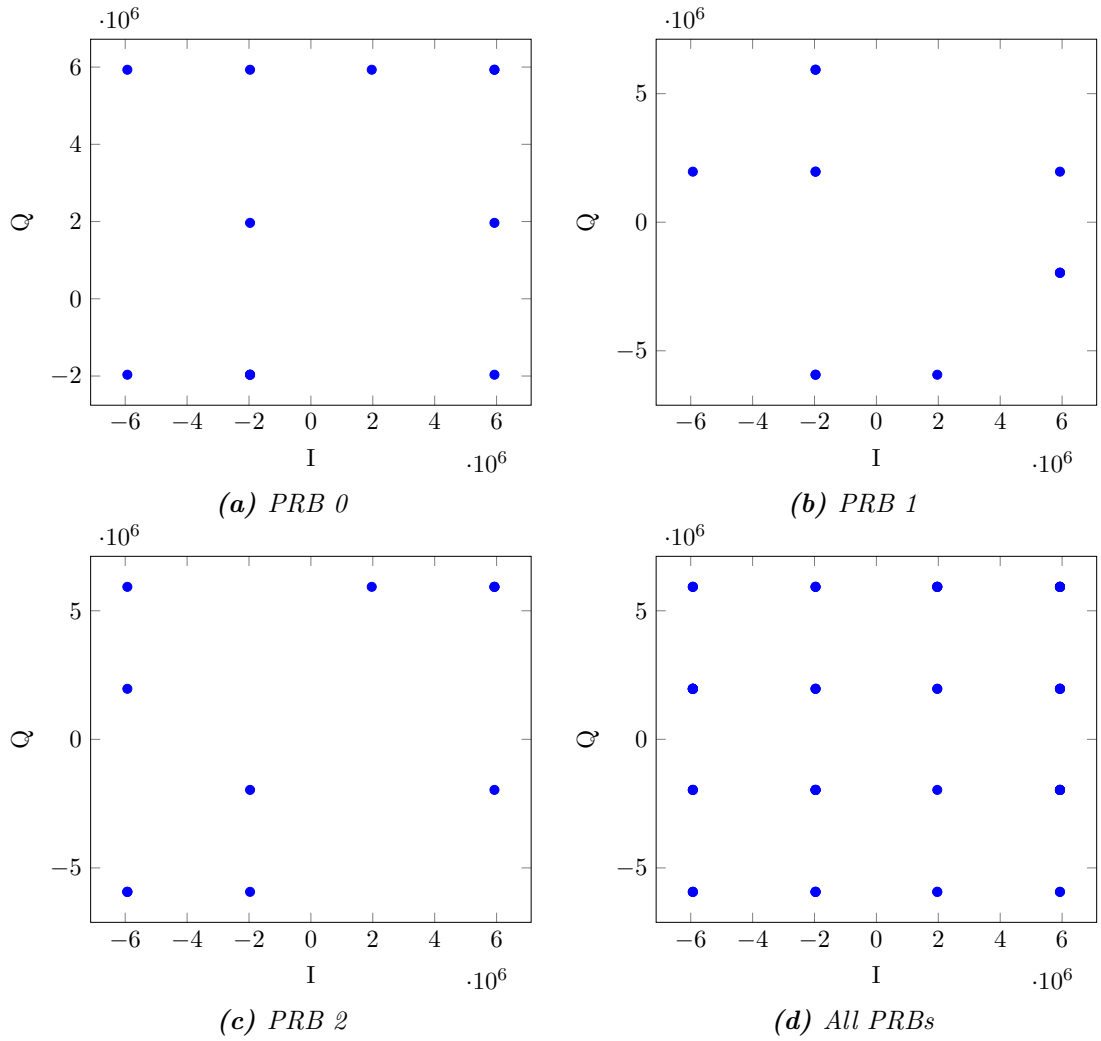


Figure 3.7. *Plot of recovered PRBs for a single numerology test case.*

Table 3.2 shows the generated and recovered IQ samples, both in their raw decompressed form and normalized with respect to average unit power. As we can see, the raw forms are identical in both cases. The normalized IQ samples are nearly the same and their slight differences can be attributed to rounding differences. The constellations seen on Fig. 3.7

reflect the raw data since they are identical in both the generated and recovered data. The normalization procedure does not impact the final result on the model since they are specific to the IP where the AXI4 signals are writting the payload symbols to.

Table 3.2. *U-plane recovered and generated data for single numerology*

	Generated				Recovered			
PRB	I raw	Q raw	I	Q	I raw	Q raw	I	Q
0	5931008	-1966080	0.71	-0.23	5931008	-1966080	0.707	-0.234
	-1966080	-1966080	-0.23	-0.23	-1966080	-1966080	-0.234	-0.234
	-5931008	5931008	-0.71	0.71	-5931008	5931008	-0.707	0.707
	5931008	1966080	0.71	0.23	5931008	1966080	0.707	0.234
	5931008	5931008	0.71	0.71	5931008	5931008	0.707	0.707
	-1966080	1966080	-0.23	0.23	-1966080	1966080	-0.234	0.234
	5931008	5931008	0.71	0.71	5931008	5931008	0.707	0.707
	1966080	5931008	0.23	0.71	1966080	5931008	0.234	0.707
	-5931008	-1966080	-0.71	-0.23	-5931008	-1966080	-0.707	-0.234
	-1966080	-1966080	-0.23	-0.23	-1966080	-1966080	-0.234	-0.234
	-1966080	5931008	-0.23	0.71	-1966080	5931008	-0.234	0.707
	-1966080	-1966080	-0.23	-0.23	-1966080	-1966080	-0.234	-0.234
1	-1966080	5931008	-0.23	0.71	-1966080	5931008	-0.234	0.707
	5931008	-196608	0.71	-0.23	5931008	-1966080	0.707	-0.234
	-1966080	-593100	-0.23	-0.71	-1966080	-5931008	-0.234	-0.707
	-5931008	1966080	-0.71	0.23	-5931008	1966080	-0.707	0.234
	-1966080	5931008	-0.23	0.71	-1966080	5931008	-0.234	0.707
	-1966080	1966080	-0.23	0.23	-1966080	1966080	-0.234	0.234
	5931008	-196608	0.71	-0.23	5931008	-1966080	0.707	-0.234
	5931008	-196608	0.71	-0.23	5931008	-1966080	0.707	-0.234
	1966080	-593100	0.23	-0.71	1966080	-5931008	0.234	-0.707
	-1966080	-593100	-0.23	-0.71	-1966080	-5931008	-0.234	-0.707
	5931008	196608	0.71	0.23	5931008	1966080	0.707	0.234
	-1966080	1966080	-0.23	0.23	-1966080	1966080	-0.234	0.234
2	5931008	-1966080	0.71	-0.23	5931008	-1966080	0.707	-0.234
	5931008	5931008	0.71	0.71	5931008	5931008	0.707	0.707
	-1966080	-1966080	-0.23	-0.23	-1966080	-1966080	-0.234	-0.234
	5931008	5931008	0.71	0.71	5931008	5931008	0.707	0.707
	-5931008	5931008	-0.71	0.71	-5931008	5931008	-0.707	0.707
	-5931008	-5931008	-0.71	-0.71	-5931008	-5931008	-0.707	-0.707
	-5931008	-5931008	-0.71	-0.71	-5931008	-5931008	-0.707	-0.707
	-5931008	-5931008	-0.71	-0.71	-5931008	-5931008	-0.707	-0.707
	-1966080	-5931008	-0.23	-0.71	-1966080	-5931008	-0.234	-0.707
	-5931008	1966080	-0.71	0.23	-5931008	1966080	-0.707	0.234
	5931008	5931008	0.71	0.71	5931008	5931008	0.707	0.707
	1966080	5931008	0.23	0.71	1966080	5931008	0.234	0.707

Compression Type Performance

We encapsulated the test data seen previously in Fig. 3.5 in bit lengths from 2 bits until 16 bit both in block-floating point and uncompressed formats. Each packet contained 13 frames that all together carried the 273 PRBs that were used as input data. Then, we ran 300 simulations for each bit length and compression type and took an average of the execution time for each case. This yielded to a total of 9000 simulations for the single numerology tests.

Despite the oRAN standard [26] presenting different compression types, block-floating point was more interesting for this work due to its simplicity and how common is found in telecommunication systems. Moreover, block floating-point emulates floating-point processing on a fixed-point processor by combining a number of signal samples with a single common exponent. It reduces overhead (compared to floating-point) but it increases quantization noise [13], [9].

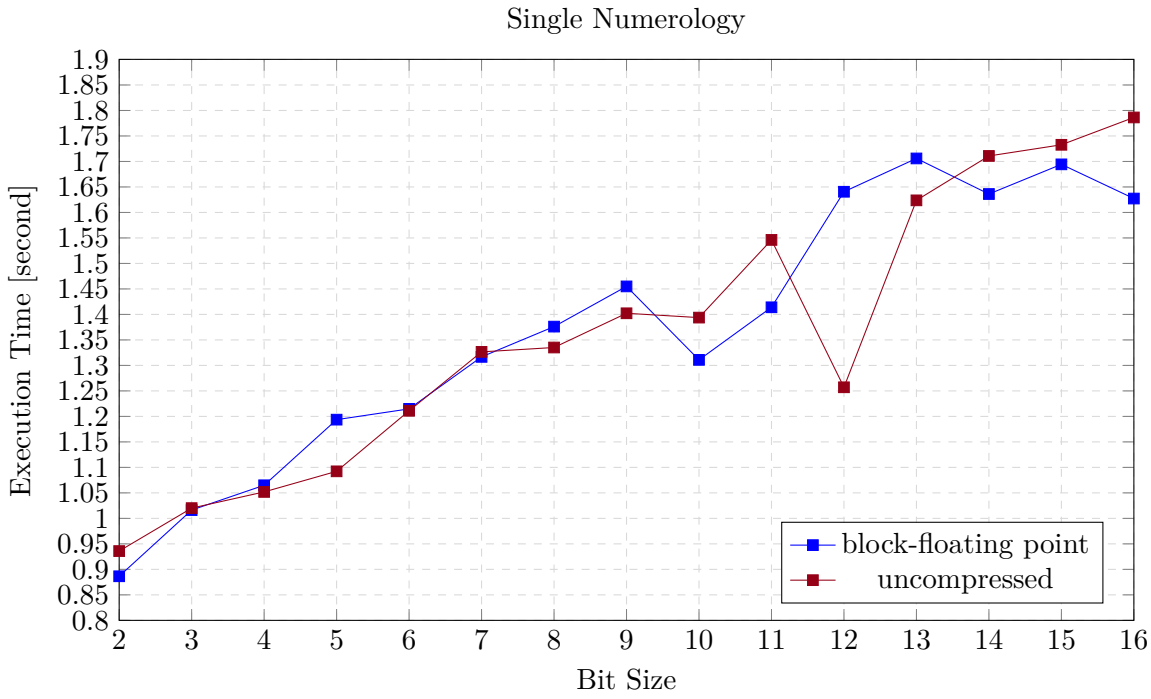


Figure 3.8. *Compression type and bit length average time comparison for the single numerology test cases*

Fig. 3.8 presents the average execution time, of the 300 simulations, depending on the bit size and compression type. The processing times for both compression types are very close to each other on the lower bit sizes up until 8bit and then they diverge there after. The biggest difference in the execution time occurs when 12 uncompressed bits are used, which has a 0.45 second difference. This timing analysis is useful to see when, for example, the data resolution is an important factor in the transmission. In this case, using 12 bit uncompressed data is the highest bit size that would offer better timing if the data resolution

is to be kept intact. If for example the user is interested in carrying the most amount of data bits and can compromise data resolution, using 10bit bfp could be a viable option since it fits the most amount of compressed bits at the lowest execution time. Overall, the uncompressed timings are lower for each bit length than its block-floating point version. This was expected since bfp adds overhead on each PRB by adding a shared exponent which represents 273 extra bytes to process in comparison with its uncompressed counterpart.

There are proposed solutions in the literature to deal with compression overheard, for example implementing an encoded common exponent as proposed in [8] or an optimized bfp arithmetic accelerator with optimized rounding and shifting operation based quantization schemes that reduce the mantissa bit length efficiently as in [22]. On the other hand, compressed data can convey the same amount of data as higher uncompressed bit sizes. For example, we can encapsulate 15bit uncompressed bytes, which takes 1.75 seconds, into a 8bit bfp frames, which takes 1.45 seconds.

Depending on how drastic the bit compression is there could be lost in resolution or precision, but for transmitting IQ data over fronthaul there could be more relaxed requirements, for example in low constellation orders QAM, such as BPSK and QPSK, or higher SNR values since they are transmitted over fiber-optic links. Moreover, compressed frames offer digital power advantages since they require less power to be transmitted and recover. This aspect will be looked at in section .3.3.5. We can also see in Fig. 3.8 that in some instances the same bit magnitude is indeed faster when compressed, this could be due to the data alignment which might require more processing for those particular bit sizes.

3.3.2 Mixed Numerology Results

C-Plane Processed Data

Mixed-numerology messages consisted of three c-plane messages, one for each numerology definition. Table 3.3 shows the generated and recovered values for the different parameters in the message and they are both identical. All messages belong to the same antenna port, which is represented by the rtcId parameter and is the same for all in this case.

Table 3.3. *c-plane messages for mixed numerology*

Parameter	Generated			Recovered		
Message	1	2	3	1	2	3
rtcId	0x0001	0x0001	0x0001	0x0001	0x0001	0x0001
SequenceID	0x0080	0x0180	0x0280	0x0080	0x0180	0x0280
FrameId	0	0	0	0	0	0
SubframeId	0	0	0	0	0	0
SlotId	0	0	0	0	0	0
startSymbolid	0	0	0	0	0	0
noSections	1	1	1	1	1	1
sectionType	3	3	3	3	3	3
timeOffset	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
frameStructure	0x00	0xC1	0xB2	0x00	0xC1	0xB2
cpLength	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
udCompHdr	0x91	0x91	0x91	0x91	0x91	0x91
sectionId	0	1	2	0	1	2
rb	0	0	0	0	0	0
symInc	0	0	0	0	0	0
startPrbc	0	180	25	0	180	25
numPrbc	50	50	50	50	50	50
reMask	0xFFF	0xFFF	0xFFF	0xFFF	0xFFF	0xFFF
numSymbol	14	14	14	14	14	14
ef	0b0	0b0	0b0	0b0	0b0	0b0
beamId	0	0	0	0	0	0
FreqOffset	0XFFE6B0	0XFFF358	0XFFF9AC	0XFFE6B0	0XFFF358	0XFFF9AC

The sequence ID allows us to understand the order on which the messages should be interpreted. If we take a look at the most significant nibble for the three cases, we can see that it increases by 0x1 on each message. That means that message 1, which has a value of 0x00, corresponds to the first message, message 2 has a value of 0x01 which is the second message and message 3 has a value of 0x2 which is the third message. The message field is not part of the frame but only part of the table in order to make it easier to read. The sequenceId must be used always to determine the order on which the messages should be arranged. The frameId, subframeId, slotId and startsymbolId are identifiers and are all set to zero. The sectionType, as discussed in the previous subsection, is of type3

which corresponds to the mixed numerology frame structure; its layout inside the frame can be seen in Fig. 2.12. The timeOffset in all cases is set to zero, which means that the time offset from the start of the slot to the start of the cyclic prefix is equal to zero samples.

The frameStructure field tells us what is the subcarrier spacing and FFT size that is used in the current message. It is different for all three cases which is expected since there were three different numerologies generated. In order to decipher what this parameter represents, we must refer to Table 2.6 and see the bit allocation for the message in order to understand the mapping of the bit definitions on each nibble. Message 1 has a frameStructure value of 0x00, which represents an FFT size of 0b0000 and a subcarrier spacing of 0b0000. According to Table. 2.7, a value of 0b0000 maps to a reserved FFT/iFFT value and according to Table. 2.8 a subcarrier spacing bit value of 0b0000 maps to: a subcarrier spacing of 15kHz and 1 slot per 1 ms subframe. Message 2 has a frameStructure value of 0xC1 which represents a bit allocation of 0b1100 for the FFT size field and a subcarrier spacing value of 0b0001. Table. 2.7 and Table. 2.8 show that the respective mapped values correspond to an FFT/iFFT size of 4096 and a subcarrier spacing of 30kHz. Finally, Message 3, has a value of 0xB2 which corresponds to a mapped FFT/iFFT and subcarrier spacing value of 2048 and 60kHz respectively.

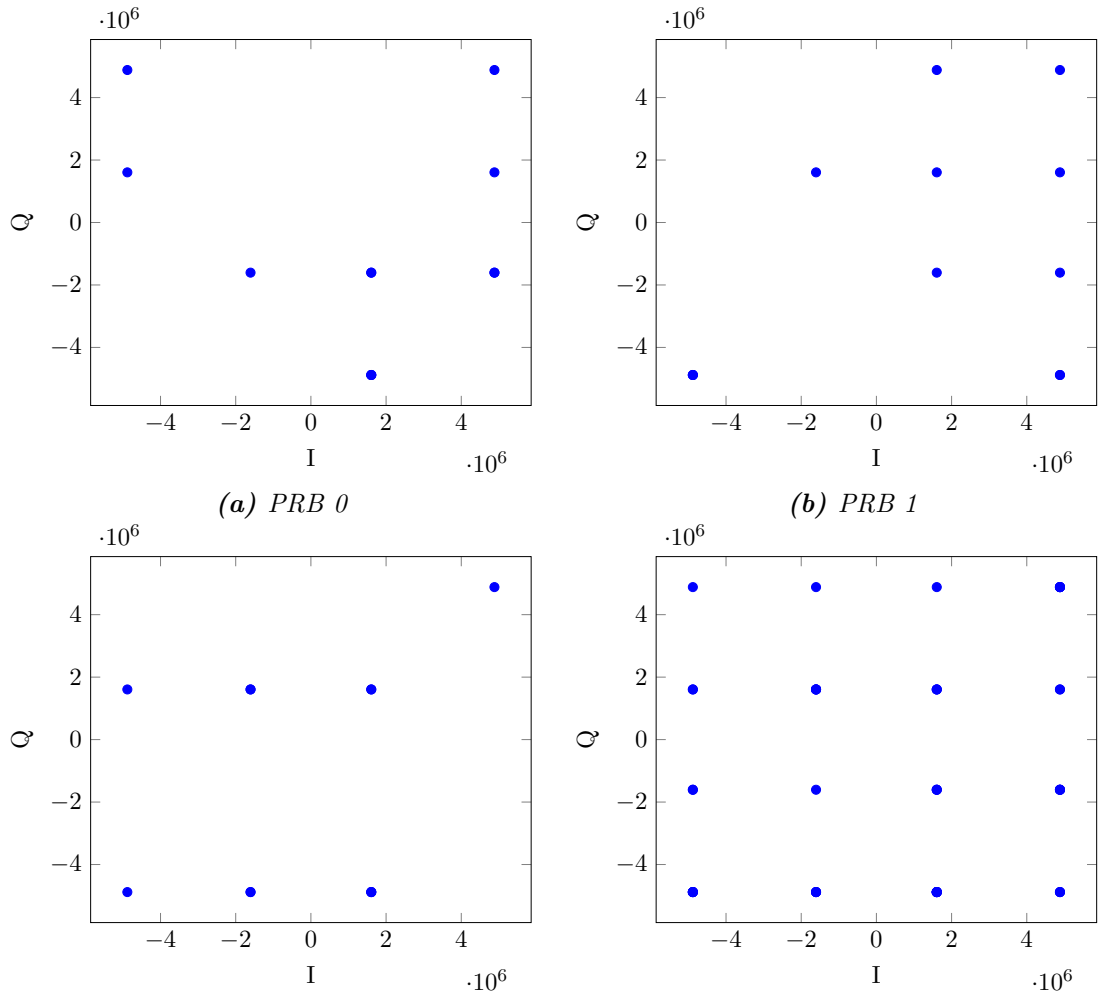
The three messages have a cpLength of zero and a 9bit block floating point compression, as specified by the udCompHdr parameter (see Table 2.9 for more information). The startPrbc field indicates the beginning PRB of the current transmission for the current numerology. The transmission begins on PRB 0, 180 and 25 for each of the messages and each message transmits 50 PRBs as specified by the numPrbc parameter. Finally, the BeamId, ef and reMask have the same values as in the single numerology case, where they represent the applicable PRBs inside the resource allocation and also the beamforming weights. All PRBs specified are enabled in this case and no beamforming was used, which agrees with the values in the table.

We can make a comparison with the deciphered data and with the channel allocations on Fig 3.6 to make sure that the data maps correctly. All messages carry 50 PRBs and they have different starting positions depending on the numerology: 0, for 15Hz; 25, for 60kHz and 180 for 30kHz. We can scale down the different values for different numerologies so that we can see what they represent for each case. For scaling all the values to $\mu = 0$, we must scale the values corresponding to 30kHz by 2 and the values corresponding to 60kHz by 4. Starting PRB for 30KHz is 180 and it transmits 50 PRBs, which means that data is sent from PRB 180 to PRB 229. Scaling those PRB values by two, we obtain a mapped range of PRB 360 to PRB 458. For 60KHz, the PRB range goes from 25 to 74 which, scalling it by 4, maps to 100 to 296. These scaled PRB ranges agree with Fig. 3.6 when $\mu = 0$. For $\mu = 1$, the PRB ranges must be scaled by 1/2 and by 2 for the 15kHz and 60kHz respectively and will yield to the new scaled ranges of: 0,...,25 for 15kHz, 50,...,148 for

60kHz and 180,...,229 for 30kHz. Lastly, For μ = 1, the PRB ranges must be scaled to 1/4 and 1/2 for 15kHz and 30kHz, which will result in: 0,...,12 for 15KHz, 25,...,74 for 60kHz and 90,...,114 for 30kHz. All new ranges match the expected channel allocation found in Fig. 3.6.

U-Plane Processed Data

Fig. 3.9 shows plots of the raw recovered data for a subcarrier spacing of 15kHz. Fig. 3.9a shows the constellation that belongs to PRB0 after it was decompressed but not normalized. As expected, the PRB does not contain all the points of the constellation but some which can vaguely resemble a 16-QAM constellation. There are more obvious points seen in Fig. 3.9b but there are still many missing. Finally, Fig. 3.9d has a plot of all the 50 PRBs found in the u-plane message which contains all the 16 symbols in the QAM constellation.



(a) PRB 0 *(b) PRB 1*

(c) PRB 2 *(d) All PRBs*

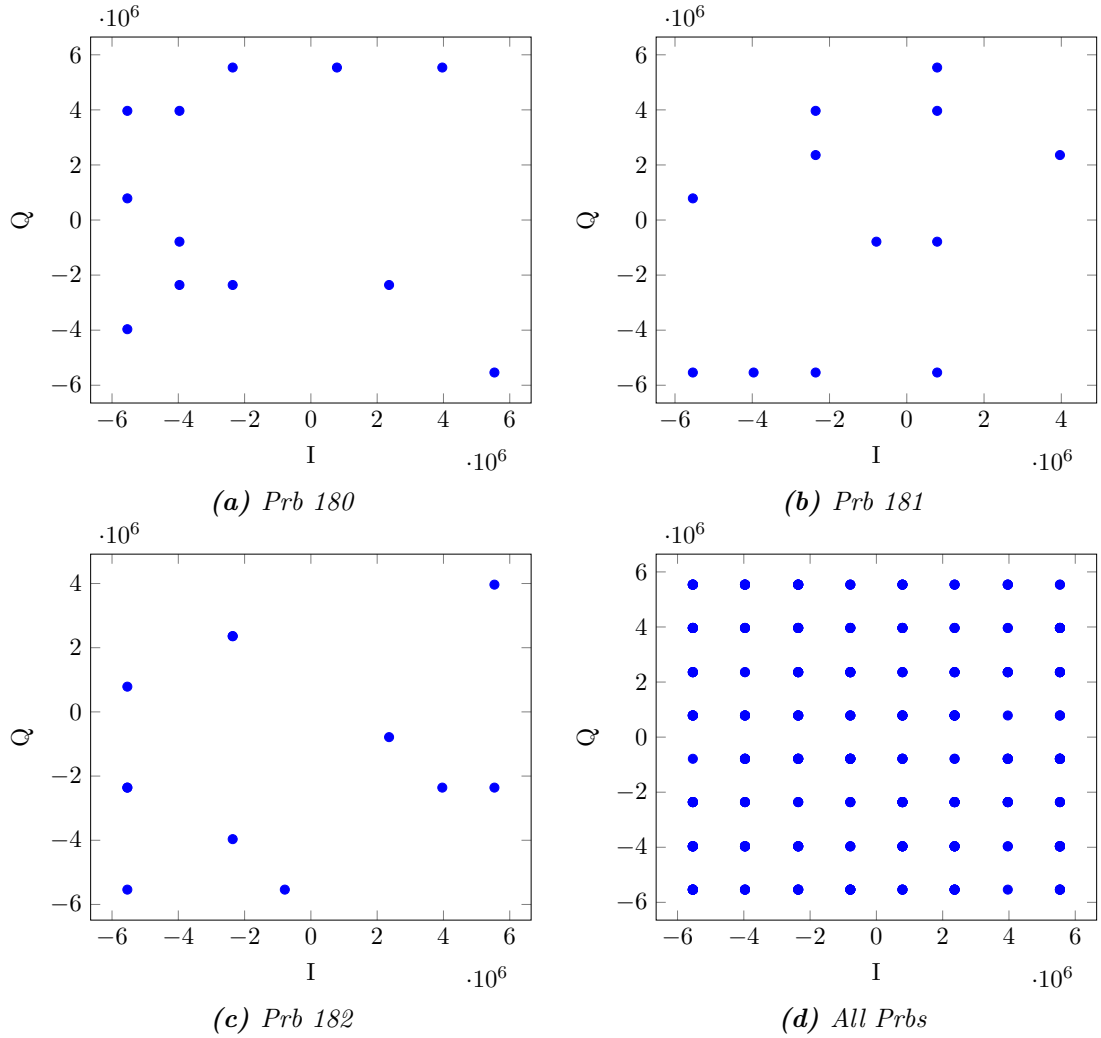
Figure 3.9. *Plot of recovered PRBs for scs=15kHz in a mixed-numerology transmission*

Table 3.4. *U-plane recovered and generated data for mixed numerology, SCS = 15kHz*

	Generated				Recovered			
PRB	I raw	Q raw	I	Q	I raw	Q raw	I	Q
0	-4882432	4882432	-0.58	0.58	-4882432	4882432	-0.707	0.707
	4882432	1605632	0.58	0.19	4882432	1605632	0.707	0.232
	1605632	-1605632	0.19	-0.19	1605632	-1605632	0.232	-0.232
	1605632	-1605632	0.19	-0.19	1605632	-1605632	0.232	-0.232
	1605632	-4882432	0.19	0.58	1605632	-4882432	0.232	-0.707
	-1605632	-1605632	-0.19	-0.19	-1605632	-1605632	-0.232	-0.232
	4882432	-1605632	0.58	-0.19	4882432	-1605632	0.707	-0.232
	4882432	-1605632	0.58	-0.19	4882432	-1605632	0.707	-0.232
	-4882432	1605632	-0.58	0.19	-4882432	1605632	-0.707	0.232
	1605632	-4882432	0.19	-0.58	1605632	-4882432	0.232	-0.707
	4882432	4882432	0.58	0.58	4882432	4882432	0.707	0.707
	1605632	-4882432	0.19	-0.58	1605632	-4882432	0.232	-0.707
1	-1605632	1605632	-0.19	0.19	-1605632	1605632	-0.232	0.232
	1605632	-1605632	0.19	-0.19	1605632	-1605632	0.232	-0.232
	1605632	4882432	0.19	0.58	1605632	4882432	0.232	0.707
	4882432	-1605632	0.58	-0.19	4882432	-1605632	0.707	-0.232
	4882432	-4882432	0.58	-0.58	4882432	-4882432	0.707	-0.707
	-4882432	-4882432	-0.58	-0.58	-4882432	-4882432	-0.707	-0.707
	-4882432	-4882432	-0.58	-0.58	-4882432	-4882432	-0.707	-0.707
	1605632	1605632	0.19	0.19	1605632	1605632	0.232	0.232
	4882432	1605632	0.58	0.19	4882432	1605632	0.707	0.232
	4882432	4882432	0.58	0.58	4882432	4882432	0.707	0.707
	4882432	-4882432	0.58	-0.58	4882432	-4882432	0.707	-0.707
	-4882432	-4882432	-0.58	-0.58	-4882432	-4882432	-0.707	-0.707
2	-4882432	-4882432	-0.58	-0.58	-4882432	-4882432	-0.707	-0.707
	-1605632	1605632	-0.19	0.19	-1605632	1605632	-0.232	0.232
	1605632	-4882432	0.19	-0.58	1605632	-4882432	0.232	-0.707
	-1605632	-4882432	-0.19	-0.58	-1605632	-4882432	-0.232	-0.707
	-1605632	1605632	-0.19	0.19	-1605632	1605632	-0.232	0.232
	1605632	1605632	0.19	0.19	1605632	1605632	0.232	0.232
	-4882432	1605632	-0.58	0.19	-4882432	1605632	-0.707	0.232
	4882432	4882432	0.58	0.58	4882432	4882432	0.707	0.707
	1605632	-4882432	0.19	-0.58	1605632	-4882432	0.232	-0.707
	1605632	1605632	0.19	0.19	1605632	1605632	0.232	0.232
	1605632	-4882432	0.19	-0.58	1605632	-4882432	0.232	-0.707
	-1605632	-4882432	-0.19	-0.58	-1605632	-4882432	-0.232	-0.707

Table 3.4 shows all the generated raw and normalized IQ symbols as well as the recovered raw and normalized IQ symbols. The raw symbols in both cases match identically. However, there is a more noticeable discrepancy between the normalized symbols. This difference can be attribute to the normalization factor that oRAN Studio (by Keysight) uses compared to the peak average power normalization used in Matlab. The fact that

the normalized symbols are slightly different does not represent an error in the processing of the frames or in the behaviour of the model since the same raw symbols that were generated were written through the AXI4 transaction.



(a) Prb 180 *(b) Prb 181*

(c) Prb 182 *(d) All Prbs*

Figure 3.10. *Plot of recovered PRBs for scs=30kHz in a mixed-numerology transmission*

Fig. 3.10 shows the recovered raw data for a transmission using a subcarrier spacing of 30kHz. As expected, in Fig 3.10a, only 12 IQ symbols can be seen on the plot representing PRB180. Since the QAM constellations is higher, it becomes extremely hard to recognize with only 1 PRB what constellation it has. Even in concurrent plots, Fig. 3.10a and 3.10a, the symbols are scattered too far apart for it to be recognizable. It finally becomes clear once all the PRBs are plotted as seen on Fig 3.10d.

Table 3.5 contains the data for all generated and recovered IQ symbols. It is important to note that the PRB number, starting from 180, matches the expected allocation provided in the c-plane message discussed earlier on this section. In order for the recovered data to belong to the specified numerology, 30kHz in this case, the reading of the written AXI4 transaction must correspond to the PRB range specified. The IQ raw data matches per-

fectly with the generated data. Moreover, there is still some slight difference among the generated and recovered normalized IQ data. This difference does not affect the accuracy of the system at all since the scaling is done using the same factors and same relationships in the real hardware implementation. The slight discrepancy seen on Table 3.5 is due to a different scaling factor used in different analysis tools.

Table 3.5. *U-plane recovered and generated data for mixed numerology, SCS = 30kHz*

	Generated				Recovered			
PRB	I raw	Q raw	I	Q	I raw	Q raw	I	Q
180	3964928	5537792	0.47	0.66	3964928	5537792	0.506	0.707
	-5537792	-3964928	-0.66	-0.47	-5537792	-3964928	-0.707	-0.506
	5537792	-5537792	0.66	-0.66	5537792	-5537792	0.707	-0.707
	-3964928	-786432	-0.47	-0.09	-3964928	-786432	-0.506	-0.100
	786432	5537792	0.09	0.66	786432	5537792	0.100	0.707
	-5537792	3964928	-0.66	0.47	-5537792	3964928	-0.707	0.506
	-3964928	-2359296	-0.47	-0.28	-3964928	-2359296	-0.506	-0.301
	-2359296	-2359296	-0.28	-0.28	-2359296	-2359296	-0.301	-0.301
	-5537792	786432	-0.66	0.09	-5537792	786432	-0.707	0.100
	-3964928	3964928	-0.47	0.47	-3964928	3964928	-0.506	0.506
	-2359296	5537792	-0.28	0.66	-2359296	5537792	-0.301	0.707
	2359296	-2359296	0.28	-0.28	2359296	-2359296	0.301	-0.301
181	-5537792	-5537792	-0.66	-0.66	-5537792	-5537792	-0.707	-0.707
	-5537792	786432	-0.66	0.09	-5537792	786432	-0.707	0.100
	-3964928	-5537792	-0.47	-0.66	-3964928	-5537792	-0.506	-0.707
	-2359296	3964928	-0.28	0.47	-2359296	3964928	-0.301	0.506
	3964928	2359296	0.47	0.28	3964928	2359296	0.506	0.301
	786432	5537792	0.09	0.66	786432	5537792	0.100	0.707
	786432	-786432	0.09	-0.09	786432	-786432	0.100	-0.100
	786432	-5537792	0.09	-0.66	786432	-5537792	0.100	-0.707
	786432	3964928	0.09	0.47	786432	3964928	0.100	0.506
	-786432	-786432	-0.09	-0.09	-786432	-786432	-0.100	-0.100
	-2359296	2359296	-0.28	0.28	-2359296	2359296	-0.301	0.301
	-2359296	-5537792	-0.28	-0.66	-2359296	-5537792	-0.301	-0.707
182	5537792	3964928	0.66	0.47	5537792	3964928	0.707	0.506
	-5537792	-5537792	-0.66	-0.66	-5537792	-5537792	-0.707	-0.707
	-5537792	786432	-0.66	0.09	-5537792	786432	-0.707	0.100
	5537792	-2359296	0.66	-0.28	5537792	-2359296	0.707	-0.301
	-786432	-5537792	-0.09	-0.66	-786432	-5537792	-0.100	-0.707
	-5537792	-2359296	-0.66	-0.28	-5537792	-2359296	-0.707	-0.301
	2359296	-786432	0.28	-0.09	2359296	-786432	0.301	-0.100
	-2359296	2359296	-0.28	0.28	-2359296	2359296	-0.301	0.301
	-5537792	-2359296	-0.66	-0.28	-5537792	-2359296	-0.707	-0.301
	3964928	-2359296	0.47	-0.28	3964928	-2359296	0.506	-0.301
	-2359296	-3964928	-0.28	-0.47	-2359296	-3964928	-0.301	-0.506
	-2359296	2359296	-0.28	0.28	-2359296	2359296	-0.301	0.301

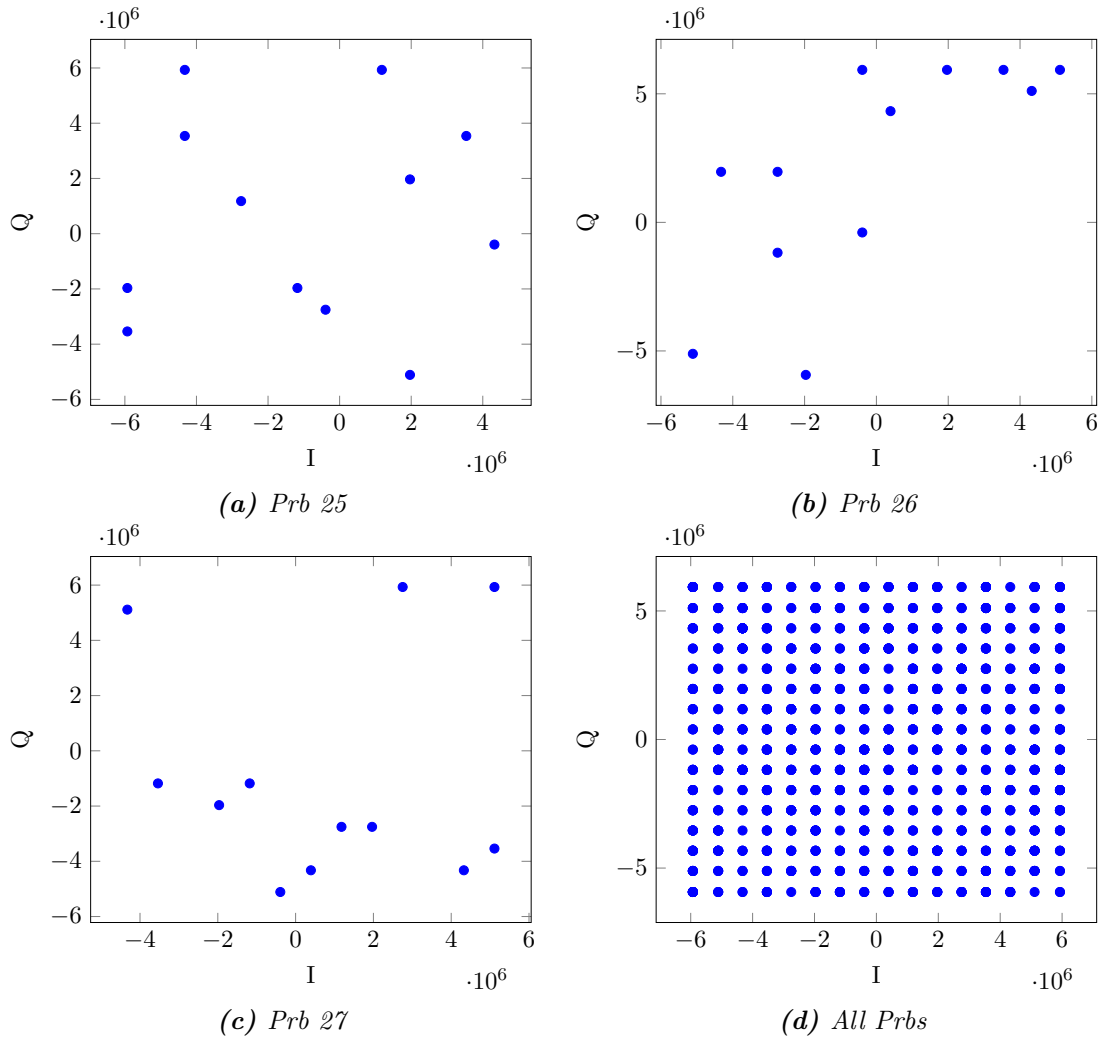


Figure 3.11. *Plot of recovered PRBs for scs=60kHz in a mixed-numerology transmission*

Finally, the constellation plots of the transmission that belong to a subcarrier spacing of 60kHz can be seen in Fig. 3.11. It follows the same patterns as in the previous two cases, where Fig. 3.11a, Fig. 3.11b and Fig. 3.11c belong to the first three PRBs where data is allocated for the 60kHz subcarrier spacing, which maps into PRBs 25, 26 and 27. In this case, the constellation is very challenging to detect by plotting only one PRB since the IQ samples are scattered in seemingly random places. As more PRBs continue to overlap, the constellation starts to become more evident, until it becomes fully plotted as seen in Fig. 3.11d. Table 3.6 shows all the generated and recovered IQ samples. As seen on the previous two cases, the raw IQ data is identical in both the generated and the recovered samples. The PRB range here starts from PRB25 and spans out until PRB74.

The three different numerology examples, each implementing a different QAM constellation, proved to process and write the payload data as it was intended. The raw representation of the decompress data was identical between the generated and the written data in all cases. The normalized samples are slightly different, in some numerologies more than in others. However, this does not represent a processing error since the scaling factors

used in the real hardware implementation is the same and does not differ from different IPs.

Table 3.6. *U-plane recovered and generated data for mixed numerology, SCS = 60kHz*

	Generated				Recovered			
PRB	I raw	Q raw	I	Q	I raw	Q raw	I	Q
25	3538944	3538944	0.42	0.42	3538944	3538944	0.482	0.482
	-4325376	5931008	-0.52	0.71	-4325376	5931008	-0.589	0.807
	-2752512	1179648	-0.22	0.14	-2752512	1179648	-0.374	0.160
	1966080	-5111808	0.23	-0.61	1966080	-5111808	0.267	-0.696
	-4325376	3538944	-0.52	0.42	-4325376	3538944	-0.589	0.482
	1179648	5931008	0.14	0.71	1179648	5931008	0.160	0.807
	-393216	-2752512	-0.05	-0.33	-393216	-2752512	-0.053	-0.374
	-1179648	-1966080	-0.14	-0.23	-1179648	-1966080	-0.160	-0.267
	1966080	1966080	0.23	0.23	1966080	1966080	0.267	0.267
	4325376	-393216	0.52	-0.05	4325376	-393216	0.589	-0.053
	-5931008	-3538944	-0.71	-0.42	-5931008	-3538944	-0.807	-0.482
	-5931008	-1966080	-0.71	-0.23	-5931008	-1966080	-0.807	-0.267
26	1966080	5931008	0.23	0.71	1966080	5931008	0.251	0.757
	3538944	5931008	0.42	0.71	3538944	5931008	0.451	0.757
	5111808	5931008	0.61	0.71	5111808	5931008	0.652	0.757
	4325376	5111808	0.52	0.61	4325376	5111808	0.552	0.652
	393216	4325376	0.05	0.52	393216	4325376	0.050	0.552
	-393216	-393216	-0.05	-0.05	-393216	-393216	-0.050	-0.050
	-2752512	-1179648	-0.33	-0.14	-2752512	-1179648	-0.351	-0.150
	-2752512	1966080	-0.33	0.23	-2752512	1966080	-0.351	0.251
	-5111808	-5111808	-0.61	-0.61	-5111808	-5111808	-0.652	-0.652
	-1966080	-5931008	-0.23	-0.71	-1966080	-5931008	-0.251	-0.757
	-4325376	1966080	-0.52	0.23	-4325376	1966080	-0.552	0.251
	-393216	5931008	-0.05	0.71	-393216	5931008	-0.050	0.757
27	-393216	-5111808	-0.05	-0.61	-393216	-5111808	-0.050	-0.652
	5111808	5931008	0.61	0.71	5111808	5931008	0.652	0.757
	5111808	-3538944	0.61	-0.42	5111808	-3538944	0.652	-0.451
	1966080	-2752512	0.23	-0.33	1966080	-2752512	0.251	-0.351
	1179648	-2752512	0.14	-0.33	1179648	-2752512	0.150	-0.351
	-4325376	5111808	-0.52	0.61	-4325376	5111808	-0.552	0.652
	2752512	5931008	0.33	0.71	2752512	5931008	0.351	0.757
	-1966080	-1966080	-0.23	-0.23	-1966080	-1966080	-0.251	-0.251
	-1179648	-1179648	-0.14	-0.14	-1179648	-1179648	-0.150	-0.150
	4325376	-4325376	0.52	-0.52	4325376	-4325376	0.552	-0.552
	393216	-4325376	0.05	-0.52	393216	-4325376	0.050	-0.552
	-3538944	-1179648	-0.42	-0.14	-3538944	-1179648	-0.451	-0.150

Compression Type Performance

The mixed numerology execution time based on bit length and compression type had less differences between the compression type compared to the single numerology test simulations. The timing average was done in the same way as it was done for the single numerology case: each bit length and compression type was ran 300 times and the average time of all the iterations per bit size was recorded. As we can see in Fig 3.12, the biggest execution time difference is seen when the simulation processed 5bit uncompressed and block-floating point data and is only 0.10 seconds.

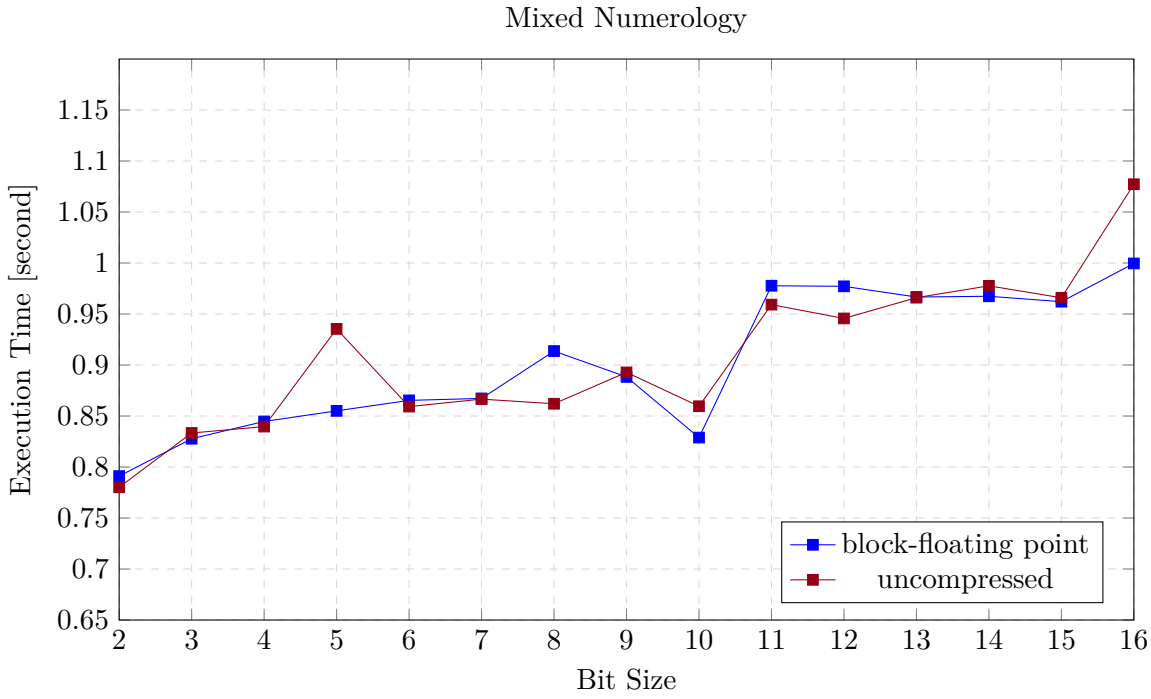


Figure 3.12. *Compression type and bit length average time comparison for the mixed numerology test cases*

The execution times does not show so much variation as they did on the single numerology case, but instead show a more intuitive raise in time as the bit length gets longer. The same analysis discussed in the previous section holds: block-floating point compressed frames can carry more information than those with the same uncompressed bit size. In this case, 10 bit block-floating point and 10 bit uncompressed frames carry the most data at faster speed. If the user is interested in better performance when it comes to speed compared to uncompressed frames or with the amount of data carried, there are other compression methods, such as dynamically adjusting the quantization bits of IQ samples according to their signal quality that are explored in [21] and adaptive frequency domain floating-point coding explored [31].

3.3.3 Ethernet Model Execution Time

Ethernet links have a well defined speed that keeps increasing as technology continues to advance. There are many factors that affect Ethernet: bit rate, propagation delay, jam time, slot time, number of hosts and minimum and maximum packet length [6]. This section presents the empirical bit rate found in the Ethernet model. The analysis to find a proper relationship between speed and frame size is explored. There are physical factors that affect the results, namely, the hardware where the simulations were executed. For example, if a more powerful computer is used or a different operating system that can handle real time computations, some of the parameters might improve or change.

Fig. 3.13 shows the same result plots presented in section 3.3.2 and section 3.3.1 but presented side-by-side. The aim is not to compare which numerology was processed faster, since both had different number of PRBs, but simply to notice a trend in time with compression type and data size. We can deduct that as the bit size increases, the execution time of the model also increases. This section seeks to find the bit rate of the model and to explore if it has any relation with frame size.

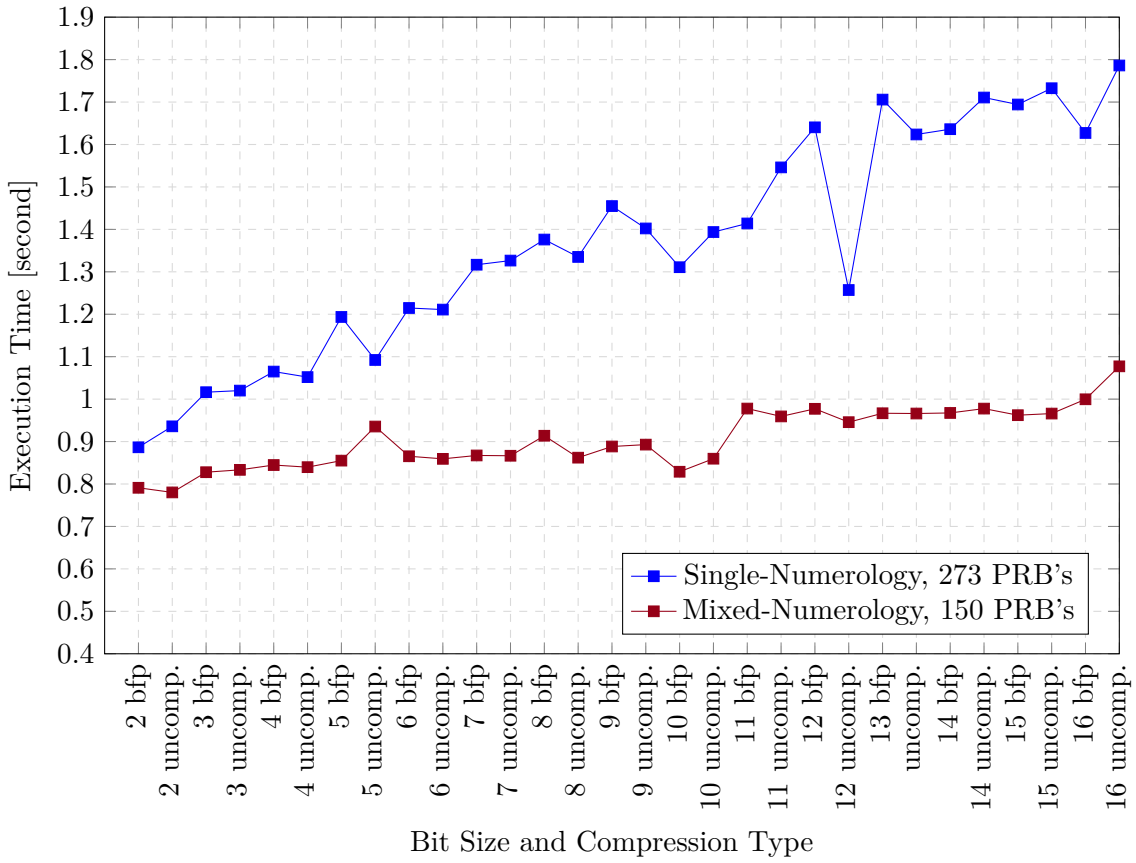


Figure 3.13. *Single and Mixed numerology execution times for different bit lengths and compression types*

In order for us to understand if there is a relationship between execution time and frame size, we created 14 frames of different sizes and ran the simulation 1000 times for each frame size in order to determine the average time that it takes. Once we obtain the time, calculating the bit rate can be found by using

$$Bitrate = \frac{Framesize}{ExecutionTime} \text{ [bits/second]} \tag{3.1}$$

where $Framesize$ is the overall size of the frame including headers and $ExecutionTime$ in our case is the average time of 1000 observations. After running the same operation for all the 14 frames, we can plot the resulting data and understand if there is a relationship between the bit rate of the frame size. The plot is seen in Fig. 3.14, where the blue scattered points are the model observations and the black line is a fitted line for the trending data.

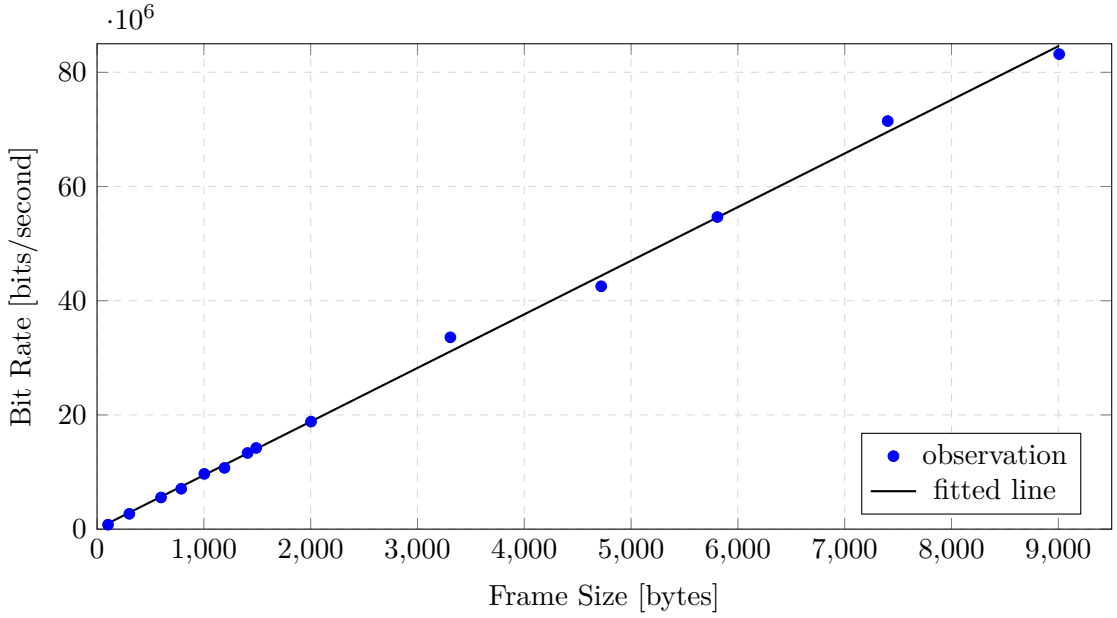


Figure 3.14. *Trending average bit rate observations based on frame size for 1000 observations per size*

According to the plot, there is a linear relation between the input frame size and the model processing bit rate. As the frames get bigger, the bit rate of the model processes data faster. This might seem counter intuitive, since one might be tempted to think that fewer bytes would be processed quicker than more bytes. However, there are processes inside the model that occur regardless of the byte size, such as instantiating the model objects, writing memory register definitions, configuring the models, etc. These wake-up routines are not very average efficient overall for processing few payload bytes. The speed relationship in the model greatly differs from physical implementations of Ethernet links where speed is not related to input frame size. However, this does not compromise the

veracity of the model since the primary scope of the model is functional and algorithmic processing rather than timing.

We can understand the timing behavior of the model more by finding a linear expression that matches the data fitted trending line so that we can get an understanding of the maximum or worst case scenario depending on the frame size. By using elementary algebra, we can find that the equation of the trending line is

$$f(x) = 9390.976x + 53300.97 \tag{3.2}$$

This will allow us to determine for example what the minimum speed, if we find the bit rate at 46 bytes, or what the maximum speed is if using 9000 byte jumbo frames. These timings are estimates and can vary depending on the hardware that is used for running the simulations but gives the user a sense of the timing relationship between different sizes.

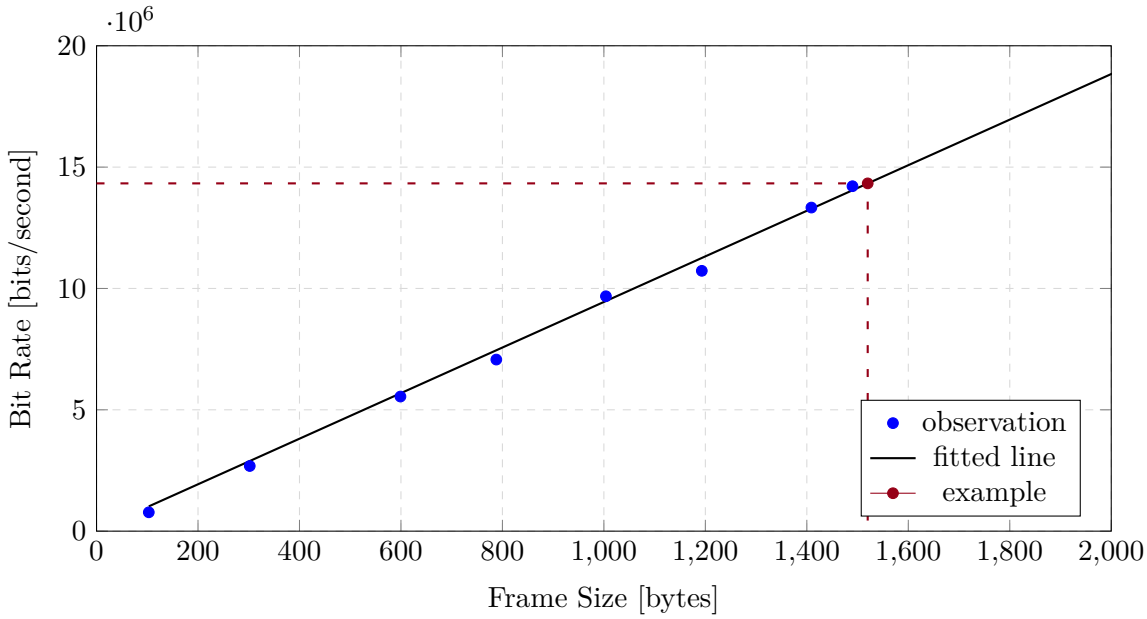


Figure 3.15. *Trending average bit rate observations, visual example bit rate for a frame size of 1518 bytes*

For our study, we will find the maximum speed of a 1500 payload byte frame, which is the maximum payload length of a regular Ethernet frame, and the maximum speed of a 9000 payload byte of a jumbo frame, which is the biggest supported size by the ethernet standard [16]. Since we are using eCPRI and oRAN encapsulated frames, then we must refer back to Fig. 2.8 and take into account how many bytes are header bytes. There is no preamble, inter-gap timing or VLAN tags in the input test cases. So the header bytes are

$$\begin{aligned}\text{Header bytes} &= \text{Dest Address} + \text{Source Address} + \text{eCPRI Type} + \text{CRC} \\ &= 6\text{bytes} + 6\text{bytes} + 2\text{bytes} + 4\text{bytes} = 18\text{bytes}\end{aligned}$$

In order to determine the proper frame size, we need to add 18 extra bytes to our desired payload size. We can estimate the bit rate value visually, by looking at Fig. 3.15 where the red dot gives an estimate of where 1518 bytes would be. If we see where that meets the y axis, we can say that the bit rate at that given size is around 14.5 Mbps(Megabits per second). If we wish to obtain a more exact result, we can make use of eq. 3.2 and plug in the desired size

$$f(1518) = 9390.976(1518) + 53300.97 = 14308802.538 \text{ bits/second} \approx 14.30 \text{ Mbps}$$

The mathematical calculation gives us more accurate values which will be used in the next section when we calculate other parameters of the model, so we will use the mathematical approach to find the maximum bitrate of a jumbo frame as well:

$$f(9018) = 9390.976(9018) + 53300.97 = 84741122.538 \text{ bits/second} \approx 84.74 \text{ Mbps}$$

Theoretically speaking, the model can potentially process any number of bytes even if beyond the size of a jumbo frame. For example several frames can be concatenated and processed as one, as long as the eCPRI and oRAN headers are appropriate. However, this does not seem to serve any meaningful purpose in the simulations. The presented bit rates in this section are limited by the hardware that is running the simulation as well as the operating system. There are studies that have been conducted, where eCPRI fronthaul processing has been implemented in an FPGA, for example in [19]. Those smaller processing modules could be used for cases where the Ethernet configuration wont change at all, but since we are interested in changing parameters and re-arranging different architectural configurations, the speed of the model is more than enough.

3.3.4 Model Execution Throughput

In this section, we will use the found parameters on the previous section in order to find the efficiency and throughput of the Ethernet model. We can interpret efficiency as the rate of useful bits that are transmitted. In other words, how many payload bits are transferred. Mathematically, we can define Ethernet efficiency as

$$\text{efficiency} = \frac{\text{payload size}}{\text{frame size}} \tag{3.3}$$

Based on eq. 3.3, we can infer that the bigger the denominator, frame overhead, compared to the useful bytes of the frame, the lower the efficiency. Ideally, both numbers should be as close as possible in order to maximize efficiency. Now we can determine the efficiency in our 1500 byte scenario

$$\text{efficiency} = \frac{1500}{1518} * 100\% = 98.814\%$$

Since we are using oRAN encapsulated eCPRI messages without VLAN tags, the overhead will always be 18 bytes. So as we increase the payload and frame size, the efficiency will increase. Fig. 3.16 shows the percentage efficiency of frame sizes all the way up to jumbo frames with 9000 bytes. In the figure, the total frame size is always 18 bytes bigger than the payload size.

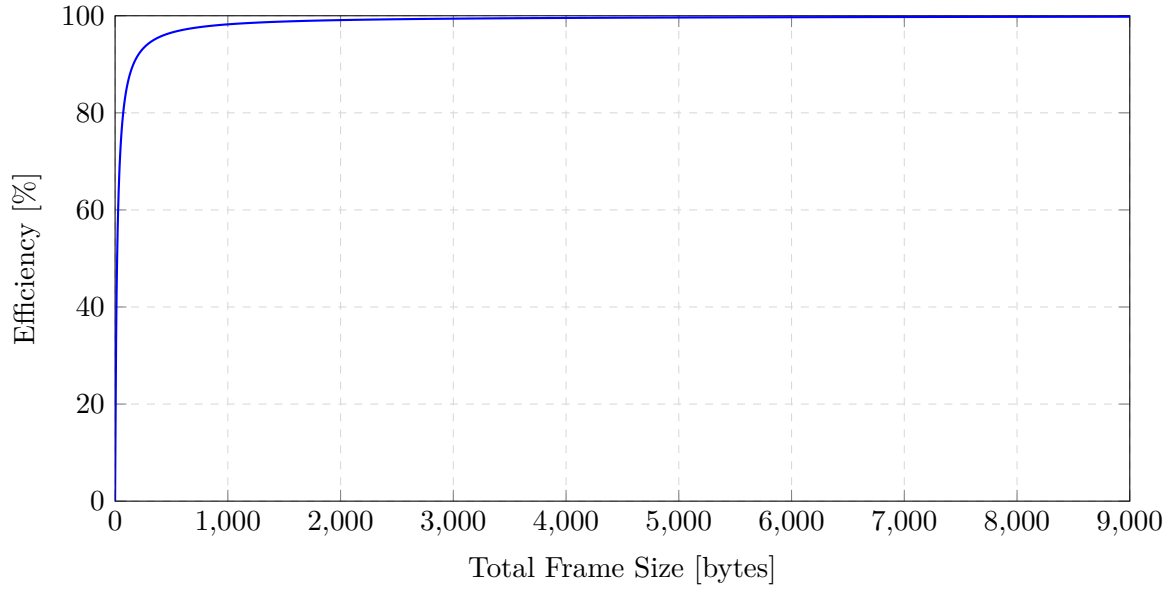


Figure 3.16. *Efficiency over multiple payload sizes of oRAN encapsualted eCPRI messages with constant 18byte headers.*

The maximum efficiency in a real regular system should be, as presented above, around 98.14%. The maximum efficiency in practise should be seen by jumbo size frames, since

that is the absolute maximum amount of bytes supported by the Ethernet standard [16]. In the case of a jumbo frame transmitting oRAN encapsulated eCPRI messages, the theoretical maximum efficiency is

$$\text{efficiency} = \frac{9000}{9018} * 100\% = 99.80\%$$

Now that we have efficiency, we can find the model's throughput which is mathematically defined as

$$\text{throughput} = \text{bit rate} * \text{efficiency} \tag{3.4}$$

Since we have found the bit rate and efficiency for the most common maximum frame size (1500 bytes) and the biggest frame size supported (9000 bytes), we can compare the throughput of the model with the throughput of a real Ethernet system. The resulting plot can be seen on Fig. 3.17

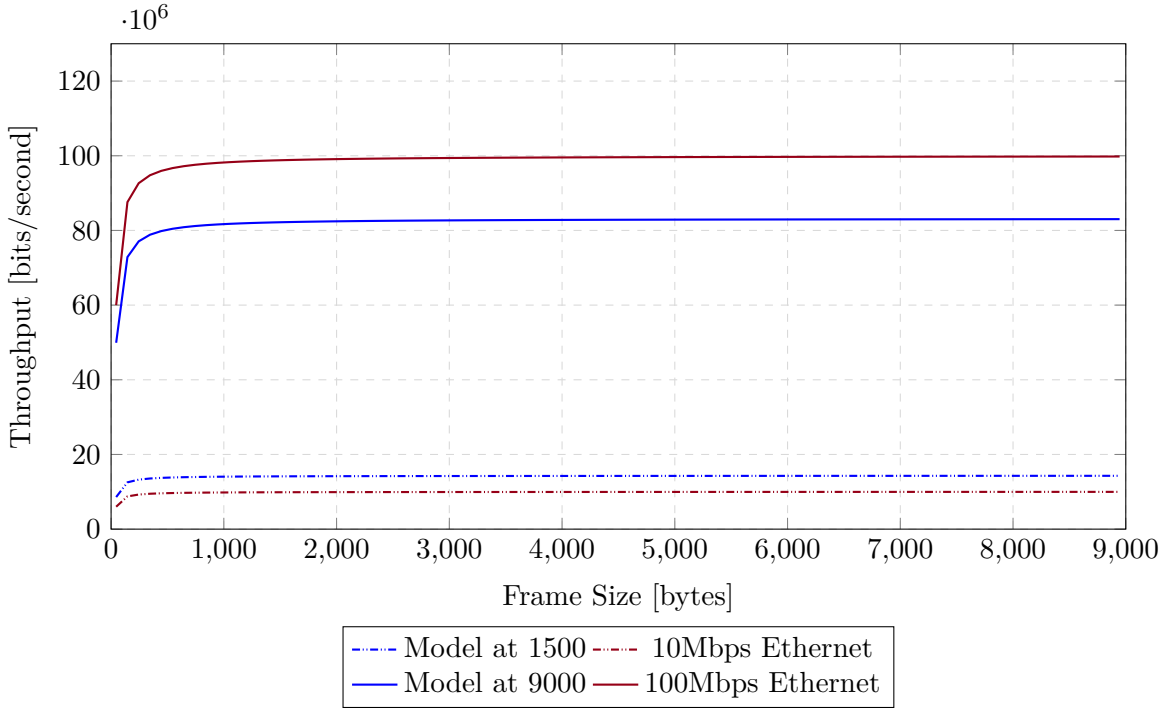


Figure 3.17. *Caption*

We can see that the throughput of the models with 1500 and 9000 bytes as payload can be compared to 10Mbps Ethernet and 100Mbps Ethernet, respectively. The throughput is constant for the whole range of acceptable sizes as it occurs in the hardware implementation. Although the speed of the model is not as fast as the real implementation, which is of course expected, it provides a noticeable resemblance when it comes to efficiency and throughput.

3.3.5 Digital Power Scaling

In this section we explore the basics of digital power scaling and use the results to provide support to the comparisons made in Sec. 3.3.1 and Sec. 3.3.2. Digital power scaling is a bigger topic for uplink transmission, since the available power of wirelessly transmitted frames is more limited. Specially if compared to downlink transmission where the base station can provide more reliable power sources.

The power level for an IQ sample transmitted over a digital interface is measured logarithmically in dBFS (dB full scale). The IQ power levels are proportional to the logarithm of $I^2 + Q^2$. The maximum power level that can be carried by one subcarrier, transporting frequency domain IQ data, is 0 dBFS. The IQ power level can be defined as

$$\begin{aligned} \text{IQ power level} &= 10log_{10}(I^2 + Q^2) - 10log_{10}(FS) \\ &= 10log_{10}(I^2 + Q^2) - 10log_{10}(FS_0 * 2^{FS\_offset}) \\ &= 10log_{10}\left(\frac{I^2 + Q^2}{FS_0 * 2^{FS\_offset}}\right) \end{aligned} \tag{3.5}$$

where the FS_offset is an M-plane parameter and FS = $\max(I^2) = \max(Q^2) = \max(I^2 + Q^2)$. If there is no FS_offset, then the expression becomes

$$\text{IQ power level} = 10log_{10}\left(\frac{I^2 + Q^2}{FS_0}\right) \tag{3.6}$$

The smallest non-zero IQ power level is defined by a parameter called the interface resolution which , when using block floating point compression, can be defined as

$$\text{Interface resolution} = -20 * log_{10}(2^{\text{Mantissa}_{\text{bits}}-1} * 2^{2^{\text{Exponent}}-1})[\text{dBFS}] \tag{3.7}$$

We can find the minimum required power for the same bit lengths and compression types that we analysed in section. 3.3.1 and section. 3.3.2. For example if we wish to find the interface resolution for 9bit block floating point

$$\begin{aligned} \text{Interface resolution} &= -20 * log_{10}(2^{9_{\text{bits}}-1} * 2^{2^4-1}) \\ &= -20 * log_{10}(256 * 32768) \\ &= -138.473 \text{ dBFS} \end{aligned} \tag{3.8}$$

and the same 9bit length but uncompressed

$$\begin{aligned} \text{Interface resolution} &= -20 * log_{10}(2^{9_{\text{bits}}-1} * 2^{2^0-1}) \\ &= -20 * log_{10}(256) \\ &= -48.164 \text{ dBFS} \end{aligned} \tag{3.9}$$

Which means that the minimum power required for detecting the IQ samples properly for a 9bit bfp IQ sample is -138.473dBFS and for a 9bit uncompressed IQ sample is -48.164dBFS. Since the maximum power level is 0 dBFS, it means that that a reading of -138.473dBFS represents 138.473dB below the maximum possible power level. This indicates that for the above 9bit case, block-floating point requires less power for detection. Moreover, is expected that the physical system will normalize the internal power of the module so that the full scale representation of 0 dBFS is properly handled.

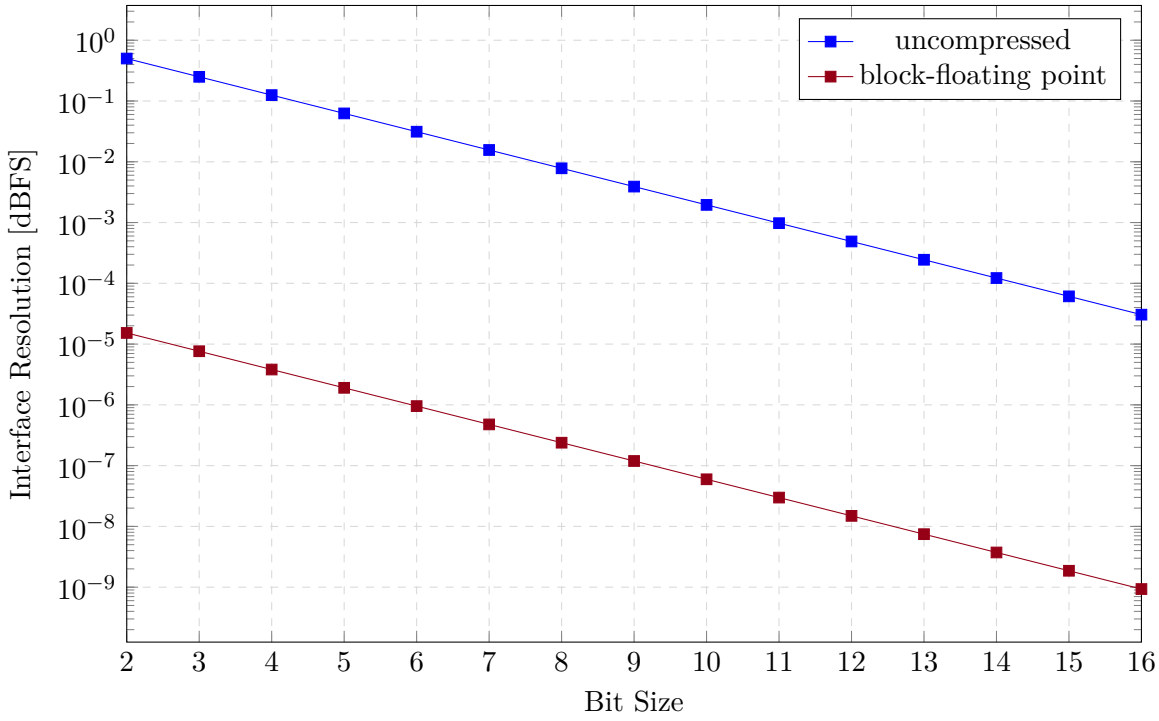


Figure 3.18. *Interface resolution depending on bit size for block floating point and uncompressed data*

Fig. 3.18 shows the different theoretical interface resolution values over all available bit lengths. We can deduce that the uncompressed data requires more power than the data that was block floating point compressed. The digital power level analysis is more interesting for UL transmission since it requires link budget and power compensation. However, this analysis shows the importance of properly choosing the compression used in fronthaul links so that the uplink recovery is possible when generating downlink data.

4 CONCLUSION

4.1 Conclusion and Results Summary

5G systems are bringing challenges that have not been faced before. 5G use cases require a reliable, fast and agile processing fronthaul connectivity that can allow for a service orientated network. These services require a smart and dynamic RAN that can allocate resources based on the needs of the end user. eCPRI and oRAN can provide a reliable and flexible solution to encapsulate IQ samples and real-time control data so that they are transported through Ethernet in the network's fronthaul. The objective of this work was to create model fronthaul Nokia subsystem IPs that transmit and process oRAN and eCPRI encapsulated IQ samples and real-time configurations through Ethernet links. All the intended goals of the work were achieved.

The main contribution of this work, is the creation of a tool that is used in the industry to design, plan and verify fronthaul connectivity modules for 5G as well as surrounding processing units found in the RRH. The model provides a scalable, flexible, bit exact configuration that has room to include new features and improvements as they are required and needed. Some of the features that are currently supported by the model are:

- Dynamic configuration for processing type2 and type0 messages.
- AXI4 memory exact output.
- One-to-one register configuration as is done in the real RTL.
- Flexible C++ source code that can be executed from different environments and frameworks.
- Scalable deployment.
- Run time system configuration.

The performance of the model was exactly as expected. The output IQ samples and configuration data are written through AXI4 interfaces either through software or to configuration files. The data type compression and bit size studies performed as expected from the practical and theoretical points of view. The execution speed and throughput are lower than the real hardware implementation, but this was expected due to the limiting factors of the hardware and operating system used to run the simulation. However, the data trends and parameters are exactly as they are in Ethernet but at a lower speed.

4.2 Future Work

Fronthaul modelling for 5G communication systems is still on is early stages as the oRAN standard is still under development. There are yet many features both in eCPRI and oRAN that are being discussed and debated on the best way to implement them in real systems. Modelling offers a very efficient solution to consider different algorithms and transport characteristics without the long developing and testing time that is required in hardware systems. There is plenty of room to carry out new research endeavours with the current implementation of the model, specially as new oRAN features start to make their way into the market.

Some interesting topics for future work would be to create a system model for uplink data. For example, implementing a model that takes into account different digital power scaling factors and creates the different gain needs that are applied to the eCPRI/oRAN frames. Also, creating a DL/UL model that expands the current system to use other compression techniques, for example modulation compression or block scaling.

REFERENCES

[1] 3GPP. *NG-RAN Architecture description*. Technical Specification (TS) 38.401. Version 15.5.0 Release 15. 3rd Generation Partnership Project (3GPP), May 2019.

[2] 3GPP. *NR and NG-RAN Overall description*. Technical Specification (TS) 38.300. Version 15.3.1 Release 15. 3rd Generation Partnership Project (3GPP), Oct. 2018.

[3] 3GPP. *Release Description*. Technical Specification (TS) 21.915. Version 15.0.0 Release 15. 3rd Generation Partnership Project (3GPP), Oct. 2019.

[4] 3GPP. *Study on new radio access technology: Radio access architecture and interfaces*. Technical Specification (TS) 38.801. Version 14.0.0. 3rd Generation Partnership Project (3GPP), Mar. 2017.

[5] S. Ahmadi. *5G NR: Architecture, Technology, Implementation, and Operation of 3GPP New Radio Standards*. Elsevier Science, 2019. ISBN: 9780128134023.

[6] D. R. Boggs, J. C. Mogul and C. A. Kent. Measured capacity of an ethernet: Myths and reality. *ACM SIGCOMM Computer Communication Review* 18.4 (1988), 222–234.

[7] D. Chitimalla, K. Kondepu, L. Valcarenghi, M. Tornatore and B. Mukherjee. 5G fronthaul-latency and jitter studies of CPRI over ethernet. *IEEE/OSA Journal of Optical Communications and Networking* 9.2 (2017), 172–182.

[8] Y. F. Choo, B. L. Evans and A. Gatherer. Complex block floating-point format with box encoding for wordlength reduction in communication systems. *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE. 2017, 1023–1028.

[9] E. L. Christensen. Block floating point for radar data. *IEEE Transactions on Aerospace and Electronic Systems* 35.1 (1999), 308–318.

[10] F. Civerchia, K. Kondepu, F. Giannone, S. Doddikrinda, P. Castoldi and L. Valcarenghi. Encapsulation Techniques and Traffic Characterisation of an Ethernet-Based 5G Fronthaul. *2018 20th International Conference on Transparent Optical Networks (ICTON)*. 2018, 1–5.

[11] E. Dahlman, S. Parkvall and J. Skold. *5G NR: The Next Generation Wireless Access Technology*. Elsevier Science, 2018. ISBN: 9780128143247.

[12] *eCPRI Interface Specification*. V2.0. Common Public Radio Interface. May 2019.

[13] D. Elam and C. Lovescu. A block floating point implementation for an N-point FFT on the TMS320C55X DSP. *Texas Instruments Application Report* (2003).

[14] M. Fiorani, S. Tombaz, J. Mårtensson, B. Skubic, L. Wosinska and P. Monti. Modeling energy performance of C-RAN with optical transport in 5G network scenarios. *Journal of Optical Communications and Networking* 8.11 (2016), B21–B34.

[15] N. J. Gomes, P. Chanclou, P. Turnbull, A. Magee and V. Jungnickel. Fronthaul evolution: From CPRI to Ethernet. *Optical Fiber Technology* (Aug. 2015).

[16] IEEE Standard for Ethernet. *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)* (2018), 1–5600.

[17] IEEE Standard for Local and metropolitan area networks – Time-Sensitive Networking for Fronthaul. *IEEE Std 802.1CM-2018* (2018), 1–62.

[18] G. Kalfas, M. Agus, A. Pagano, L. A. Neto, A. Mesodiakaki, C. Vagionas, J. Vardakas, E. Datsika, C. Verikoukis and N. Pleros. Converged analog fiber-wireless point-to-multipoint architecture for eCPRI 5G fronthaul networks. *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2019, 1–6.

[19] D. T. Kiet, T. M. Hieu, N. Q. Hung, N. Van Cuong, V. T. Van and P. N. Cuong. Research and Implementation of eCPRI Processing Module for Fronthaul Network on FPGA in 5G–NR gNodeB Base Station. *2020 4th International Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom)*. IEEE. 2020, 1–5.

[20] S. T. Le, T. Drenski, A. Hills, M. King, K. Kim, Y. Matsui and T. Sizer. 400Gb/s real-time transmission supporting CPRI and eCPRI traffic for hybrid LTE-5G networks. *Optical Fiber Communication Conference*. Optical Society of America. 2020, Th4C–4.

[21] L. Li, M. Bi, W. Wang, Y. Fu, X. Miao and W. Hu. SINR-oriented flexible quantization bits for optical-wireless deep converged eCPRI. *2018 International Conference on Optical Network Design and Modeling (ONDM)*. IEEE. 2018, 172–177.

[22] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou and X. Ji. High-performance FPGA-based CNN accelerator with block-floating-point arithmetic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.8 (2019), 1874–1885.

[23] L. Marijanović, S. Schwarz and M. Rupp. Multiplexing Services in 5G and Beyond: Optimal Resource Allocation Based on Mixed Numerology and Mini-Slots. *IEEE Access* 8 (2020), 209537–209555.

[24] A. Marotta, D. Cassioli, K. Kondepu, C. Antonelli and L. Valcarenghi. Exploiting flexible functional split in converged software defined access networks. *IEEE/OSA Journal of Optical Communications and Networking* 11.11 (2019), 536–546.

[25] *O-RAN:Towards an Open and Smart RAN*. White Paper. O-RAN Alliance. Oct. 2018.

[26] oRAN. *Control, User and Synchronization Plane Specification*. Technical Specification ORAN-WG4.CUS.0. Version 02.00. O-RAN Alliance, 2019.

[27] G. Otero Pérez, D. Larrabeiti López and J. A. Hernández. 5G New Radio Fronthaul Network Design for eCPRI-IEEE 802.1CM and Extreme Latency Percentiles. *IEEE Access* 7 (2019), 82218–82230.

[28] R. M. Rao, M. Fontaine and R. Veisllari. A Reconfigurable Architecture for Packet Based 5G Transport Networks. *2018 IEEE 5G World Forum (5GWF)*. 2018, 474–477.

[29] S. K. Singh, R. Singh and B. Kumbhani. The Evolution of Radio Access Network Towards Open-RAN: Challenges and Opportunities. *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2020, 1–6.

[30] C. Spurgeon. *Ethernet: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2000. ISBN: 9780596552824.

[31] W. Yu, J. Chen, H. Sun and W. Li. The compression of wireless base station lte-ir data using adaptive frequency domain floating-point coding. *2014 12th International Conference on Signal Processing (ICSP)*. IEEE. 2014, 1703–1708.