FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Measuring and modelling 5G RAN performance

Marija Jocic



Master's degree in Electrical and Computer Engineering

Supervisor: Ana Cristina Costa Aguiar

July 30, 2024

© Marija Jocic, 2024

Measuring and modelling 5G RAN performance

Marija Jocic

Master's degree in Electrical and Computer Engineering

July 30, 2024

Abstract

The emergence of fifth generation technology has ushered in a constantly changing era in cellular networks, marked by significant architectural shifts. This fifth generation standard sets high standards for speed, bandwidth, coverage, latency and energy usage. Notably, the adoption of Open Radio Access Networks has revolutionized radio access, allowing the virtualization and distributed deployment of network functions across diverse platforms. Another groundbreaking aspect of 5G is the advent of Non-Public Networks (NPNs). These networks are independent of public telecom operators and designed to accommodate specific users or devices within a single organization.

This dissertation extends the work on a previously implemented research NPN utilizing OpenAirInterface software, integrating open-source components, software-defined radios, and a core server. The network, established at IT Porto within FEUP, has already undergone successful initial validation. This research aims to further explore the infrastructure's capabilities and functionality, focusing on energy and power consumption, as well as the throughput. It will assess the impact of different data rates on overall energy and power consumption, and determine the maximum achievable throughput. Another stage of the work involves examining the role of processor energy and power consumption in the total energy and power consumption for different data rates. Furthermore, this study provides deeper insights into the reasons for processor energy and power consumption by investigating the CPU performance counters.

The findings of this project reveal certain potential throughput limitations and explain the underlying reasons for them. Additionally, the results indicate that there is no significant influence of throughput on power and energy consumption. Furthermore, it demonstrates that overall energy consumption is not solely dependent on the processor's energy use.

The insights gained from this project will benefit future researchers and developers working on non-public network 5G infrastructures that use virtualized Radio Access Networks. The study will help them understand the potential challenges associated with working with these types of networks and how to address them. Additionally, this research introduces new discussions on performance trade-offs and the factors influencing energy consumption, paving the way for future improvements in energy efficiency and leading to more sustainable networks. Enhanced energy efficiency means improved resource allocation, which reduces the unnecessary energy consumption. Similarly, adjusting network slicing and Quality of Service (QoS) settings to ensure that high-priority data receives the necessary bandwidth can maximize achievable throughput.

Keywords: 5G, Non-public network, Virtualized Radio Access Network, Open-source

Sustainable Development Goals (SDGs) of the United Nations

This research promotes resilient infrastructure and fosters innovation by improving 5G network technologies, making them more accessible and efficient through the use of open-source software. This can enhance connectivity and technological development in various industries, especially in developing regions. This represents a contribution to the SDG number 9: Industry, Innovation, and Infrastructure.

By advancing 5G technology, the work supports the creation of smart cities. Enhanced connectivity and network efficiency can lead to improved urban planning, smarter transportation systems, and better resource management, contributing to more sustainable urban environments. This represents a contribution to the SDG number 11: Sustainable Cities and Communities.

Focus on evaluating and optimizing energy consumption in 5G networks aligns with the goal of ensuring sustainable consumption and production patterns. By identifying energy-efficient configurations and addressing power consumption trade-offs, this work can help reduce the environmental impact of telecommunications infrastructure. This relates to SDG number 12: Responsible Consumption and Production.

Utilizing open-source software encourages collaboration and knowledge sharing among researchers, industry stakeholders, and communities. This aligns with the goal of strengthening global partnerships to support and achieve the ambitious targets of the sustainable development agenda, which is a SDG number 17: Partnerships for the Goals.

Acknowledgments

First of all, I am deeply grateful for the invaluable guidance, patience, and time provided by Ana Costa Aguiar, my supervisor throughout this research. Her mentorship was indispensable in completing this work, and I am fortunate to have had such a supportive supervisor.

I extend sincere gratitude to Sofia Cardoso Martins, my colleague in the IT team. Her assistance and insights were instrumental in swiftly grasping and resolving challenges throughout the project. I also thank José Rodrigues for introducing me to this network setup and my master's colleagues who worked with me. I am thankful for the opportunity to collaborate with them.

I express my heartfelt thanks to my family, partner, and friends for their unwavering support throughout my studies, including my master's degree. They have been my constant motivation and inspiration, and I wouldn't have reached this point without them.

Finally, I acknowledge Project Route 25, Nr. C645463824-00000063, funded by FCT/MCTES through national funds and, where applicable, co-funded by EU funds under the project UIDB/50008/2020 and by the European Union / Next Generation EU, through the Programa de Recuperação e Resiliência (PRR). I also express my gratitude to IT Porto for hosting the project.

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 1
- 1.3 Goals . . . 2
- 1.4 Document structure . . . 2

2 Background and State of the Art **3**
- 2.1 5G . . . 3
  - 2.1.1 User Equipment . . . 4
  - 2.1.2 Core Network . . . 4
  - 2.1.3 Radio Access Network . . . 6
- 2.2 Virtualized Radio Access Networks (v-RANs) . . . 8
- 2.3 Non-public networks (NPNs) . . . 9
- 2.4 State of the Art . . . 11

3 Methodology **14**
- 3.1 Problem formulation . . . 14
- 3.2 Experimental setup . . . 15
- 3.3 Open-source 5G NPN Deployment . . . 17
  - 3.3.1 Virtualized RAN deployment . . . 17
  - 3.3.2 Core Network and UE deployment . . . 19
  - 3.3.3 Network initialization and validation . . . 19
- 3.4 Experimental method . . . 21
  - 3.4.1 Power meter configuration . . . 22
  - 3.4.2 Generating data with *iperf* . . . 23
  - 3.4.3 Measuring power consumption with RAPL . . . 23
  - 3.4.4 Monitoring CPU events with *perf* . . . 24

4 Results Analysis **26**
- 4.1 Defining baselines - Initial measurements . . . 26
  - 4.1.1 Impact of core frequency scaling configuration . . . 26
  - 4.1.2 Obtaining baselines . . . 27
  - 4.1.3 Obtaining baselines when the processor is 100% active . . . 29
- 4.2 Dependency of Power Consumption on Data Rate . . . 30
  - 4.2.1 Understanding Throughput Limitations for UDP traffic . . . 33
  - 4.2.2 Understanding Throughput Limitations for TCP traffic . . . 36
- 4.3 Assessing Processor Share in Power Consumption . . . 39
  - 4.3.1 Fraction of processor power in total power . . . 39

4.3.2 Fraction of processor power in total power when processor is 100% active 41
4.3.3 Dependency of CPU's power consumption on Data Rate 42
4.4 Data rate effect on CPU performance counters 45

5 Conclusion and Future work **54**

References **55**

A Configuration files **57**
A.1 Control Unit(CU) 57
A.2 Distributed Unit(DU) 57
A.3 User Equipment (UE) 58

List of Figures

2.1 5G architecture, figure based on [10] . . . 4
2.2 Base station layers, [22] . . . 7
2.3 RAN split on CU, DU and RU, [22] . . . 8
2.4 Fully and Partially centralized RAN, [9] . . . 9
2.5 Different types of Non-Public Networks . . . 10
(a) SNPN, [1] . . . 10
(b) PNI-NPN with Shared RAN, [1] . . . 10
(c) PNI-NPN with Shared RAN and Control Plane, [1] . . . 10
(d) NPN hosted by the public network, [1] . . . 10

3.1 Testbed . . . 16
3.2 Testbed in the lab, CU, DU and USRPs . . . 16
3.3 Testbed in the lab, UE and power meter . . . 17
3.4 SoftPanel software tool . . . 22
3.5 Average, window and integration operating modes for the power meter, [8] . . . 22

4.1 Power consumption of the devices - Initial measurement . . . 27
(a) Power consumption of CU+UE . . . 27
(b) Power consumption of DU . . . 27
(c) Power consumption of DU USRP . . . 27
(d) Power consumption of UE USRP . . . 27
4.2 Power consumption of the devices - After solving DU power consumption issue . . . 28
(a) Power consumption of CU+UE . . . 28
(b) Power consumption of DU . . . 28
(c) Power consumption of DU USRP . . . 28
(d) Power consumption of UE USRP . . . 28
4.3 Power consumption of the devices - After separating UE from the CU . . . 29
(a) Power consumption of CU . . . 29
(b) Power consumption of DU . . . 29
(c) Power consumption of UE . . . 29
4.4 Total power consumption of the devices when processor is fully activated . . . 31
(a) Power consumption of CU . . . 31
(b) Power consumption of DU . . . 31
(c) Power consumption of UE . . . 31
4.6 Throughput vs excess over average OAI process power consumption for UDP traffic of 16Mb . . . 32
(a) Throughput of the core vs excess over average OAI process power consumption of DU . . . 32

(b) Throughput of the UE vs excess over average OAI process power consumption of UE . . . 32
4.7 Throughput vs excess over average OAI process power consumption for UDP traffic of 32Mb . . . 32
(a) Throughput of the core vs excess over average OAI process power consumption of DU . . . 32
(b) Throughput of the UE vs excess over average OAI process power consumption of UE . . . 32
4.5 Throughput vs excess over average OAI process consumption for UDP traffic of 8Mb . . . 33
(a) Throughput of the core vs excess over average OAI process power consumption of DU . . . 33
(b) Throughput of the UE vs excess over average OAI process power consumption of UE . . . 33
4.8 Throughput vs excess over average OAI process power consumption for UDP traffic of 64Mb . . . 33
(a) Throughput of the core vs excess over average OAI process power consumption of DU . . . 33
(b) Throughput of the UE vs excess over average OAI process power consumption of UE . . . 33
4.9 Throughput of the core and UE for UDP traffic, and the number of retransmissions in 3 rounds . . . 35
4.10 Throughput vs excess over average OAI process power consumption for UDP traffic of 32Mb while monitoring the buffer . . . 36
(a) Throughput of the core vs excess over average OAI process power consumption of DU . . . 36
(b) Throughput of the UE vs excess over average OAI process power consumption of UE . . . 36
4.11 Throughput vs excess over average OAI process power consumption for UDP traffic of 64Mb while monitoring the buffer . . . 36
(a) Throughput of the core vs excess over average OAI process power consumption of DU . . . 36
(b) Throughput of the UE vs excess over average OAI process power consumption of UE . . . 36
4.12 Throughput vs excess over average OAI process power consumption for TCP traffic 37
(a) Throughput of the core vs excess over average OAI process power consumption of DU . . . 37
(b) Throughput of the UE vs excess over average OAI process power consumption of UE . . . 37
4.13 Throughput of the core and UE for TCP traffic and the number of retransmissions 38
4.14 Throughput vs excess over average OAI process power consumption for TCP traffic when the core is at C0 state . . . 38
(a) Throughput of the core vs excess over average OAI process power consumption of DU when the core is in C0 state . . . 38
(b) Throughput of the UE vs excess over average OAI process power consumption of UE when the core is in C0 state . . . 38
4.15 Throughput of the core and UE for TCP traffic and the number of retransmissions when the core is in C0 state . . . 39

4.16 CPU power consumption of the devices . . . 40
(a) Power consumption of CU's processor . . . 40
(b) Power consumption of DU's processor . . . 40
(c) Power consumption of the UE's processor . . . 40
4.17 CPU power consumption of the devices when processor is fully activated . . . 41
(a) Power consumption of CU's processor . . . 41
(b) Power consumption of DU's processor . . . 41
(c) Power consumption of the UE's processor . . . 41
4.18 Excess over average OAI process baseline power consumption of the CU - Total vs Processor . . . 43
(a) Excess over average OAI process baseline power consumption of CU for different UDP data rates measured on power meter . . . 43
(b) Excess over average OAI process baseline power consumption of CU's CPU for different UDP data rates measured with RAPL . . . 43
4.19 Excess over average OAI process baseline power consumption of the DU - Total vs Processor . . . 43
(a) Excess over average OAI process baseline power consumption of DU for different UDP data rates measured on power meter . . . 43
(b) Excess over average OAI process baseline power consumption of DU's CPU for different UDP data rates measured with RAPL . . . 43
4.20 Excess over average OAI process baseline power consumption of the UE - Total vs Processor . . . 44
(a) Excess over average OAI process baseline power consumption of UE for different UDP data rates measured on power meter . . . 44
(b) Excess over average OAI process baseline power consumption of UE's CPU for different UDP data rates measured with RAPL . . . 44
4.21 Statistics of excess power consumption over average for the total power and processor power of the devices . . . 45
(a) CU statistics for excess power over the average for total and processor power consumption . . . 45
(b) DU statistics for excess power over the average for total and processor power consumption . . . 45
(c) UE statistics for excess power over the average for total and processor power consumption . . . 45
4.22 Power consumption of the CU - Total vs Processor . . . 46
(a) Power consumption of CU for different data rates measured on power meter 46
(b) Power consumption of CU's CPU for different data rates measured with RAPL 46
4.23 Power consumption of the DU - Total vs Processor . . . 46
(a) Power consumption of DU for different data rates measured on power meter 46
(b) Power consumption of DU's CPU for different data rates measured with RAPL 46
4.24 Power consumption of the UE - Total vs Processor . . . 46
(a) Power consumption of UE for different data rates measured on power meter 46
(b) Power consumption of UE's CPU for different data rates measured with RAPL 46
4.25 Number of cycles and Cache miss ratio per function of the CU for different data rates . . . 47
(a) Number of cycles per function of the CU for different data rates . . . 47
(b) Cache miss ratio per function of the CU for different data rates . . . 47

4.26 Number of cycles and Cache miss ratio per function of the DU for different data rates . . . 48
(a) Number of cycles per function of the DU for different data rates . . . 48
(b) Cache miss ratio per function of the DU for different data rates . . . 48
4.27 Number of cycles and Cache miss ratio per function of the UE for different data rates . . . 48
(a) Number of cycles per function of the UE for different data rates . . . 48
(b) Cache miss ratio per function of the UE for different data rates . . . 48
4.28 Number of cycles and Cache miss ratio for memory move function of the CU for different data rates . . . 49
(a) Number of cycles for memory move function of the CU for different data rates 49
(b) Cache miss ratio for memory move function of the CU for different data rates 49
4.29 Number of cycles and Cache miss ratio for GTP function of the CU for different data rates . . . 50
(a) Number of cycles for GTP function of the CU for different data rates . . . 50
(b) Cache miss ratio for GTP function of the CU for different data rates . . . 50
4.30 Number of cycles and Cache miss ratio for pthread mutex unlock function of the CU for different data rates . . . 50
(a) Number of cycles for pthread mutex unlock function of the CU for different data rates . . . 50
(b) Cache miss ratio for pthread mutex unlock function of the CU for different data rates . . . 50
4.31 Number of cycles and Cache miss ratio for memory move function of the DU for different data rates . . . 51
(a) Number of cycles for memory move function of the DU for different data rates . . . 51
(b) Cache miss ratio for memory move function of the DU for different data rates 51
4.32 Number of cycles and Cache miss ratio for nrLDPC encoding function of the DU for different data rates . . . 51
(a) Number of cycles for nrLDPC encoding function of the DU for different data rates . . . 51
(b) Cache miss ratio for nrLDPC encoding function of the DU for different data rates . . . 51
4.33 Number of cycles and Cache miss ratio for memory move function of the UE for different data rates . . . 52
(a) Number of cycles for memory move of the UE for different data rates . . . 52
(b) Cache miss ratio for memory move function of the UE for different data rates 52
4.34 Number of cycles and Cache miss ratio for New Radio LDCP function of the UE for different data rates . . . 52
(a) Number of cycles for New Radio LDCP function of the UE for different data rates . . . 52
(b) Cache miss ratio for New Radio LDCP function of the UE for different data rates . . . 52
4.35 Number of cycles and Cache miss ratio for New Radio LDCP decoding function of the UE for different data rates . . . 53
(a) Number of cycles for New Radio LDCP decoding function of the UE for different data rates . . . 53

(b) Cache miss ratio for New Radio LDCP decoding function of the UE for different data rates 53

List of Tables

2.1 Open-source core network frameworks 6
2.2 Open-source RAN frameworks 9
2.3 Future work regarding state of the art 13

3.1 Physical layer parameter configuration 17
3.2 Hardware specification 18

4.1 Average power and energy consumption of the devices for three different baselines 28
4.2 Processor C-states, from [27] 30
4.3 Average total power consumption and energy consumption of the devices for three different cases when processor is fully activated 30
4.4 Average power and energy consumption of the devices for different rates of UDP traffic 32
4.5 Average achieved throughput and standard deviation at the server side for different rates of UDP traffic on the client side 33
4.6 Average power and energy consumption of the devices for TCP traffic 37
4.7 Average power and energy consumption of the devices for TCP traffic when the core is in C0 state 38
4.8 Average processor power and energy consumption of the devices for three different cases 40
4.9 Percentage of processor power consumption in total power consumption 40
4.10 Average processor power and energy consumption of the devices for three different cases when processor is fully activated 42
4.11 Percentage of processor power consumption in total power consumption when processor is fully activated 42
4.12 Average processor energy consumption of the devices for different data rates of UDP traffic 43
4.13 Percentage of power consumption in total power consumption for different data rates of UDP traffic 44

Abbreviations and Symbols

5G	Fifth Generation
5GC	5G Core
ADC	Analog to digital conversion
AF	Application Function
AMF	Access and Mobility Management Function
AR	Augmented Reality
ARQ	Automatic Repeat Request
AUSF	Authentication Support Function
BBU	BaseBand Unit
BS	Base Station
BSF	Binding Support Function
BTS	Base Transceiver Station
CN	Core Network
CP	Control Plane
CPU	Central Processing Unit
CS	Cloud Site
CU	Centralized Unit
DAC	Digital to Analog Conversion
DN	Data Network
DNN	Data Network Names
DRAM	Dynamic Random Access Memory
DU	Distributed Unit
eMMB	enhanced Mobile Broadband
ETSI	European Telecommunications Standards Institute
FFT	Fast Fourier Transform
GPP	General Purpose Processors
GPRS	General Packet Radio Service
GTP	GPRS Tunneling Protocol
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
IP	Internet Protocol
MAC	Medium Access Control
MCC	mobile Country Code
MCS	Modulation and Coding Scheme
mMTC	massive Machine-Type Communications
MNC	Mobile Newtork Code
NEF	Network Exposure Function
NFV	Network Function Virtualization
NID	Network IDentifier
NPN	Non-Public Network
NR	New Radio
NRF	Network Repository Function

NSSF	Network Slicing Selection Function
NSSAAF	Network Slice and Service Authorization and Accounting Function
LADN	Local Area Data Network
OAI	OpenAirInterface
OPc	Operator Key
OS	Operating System
PCF	Policy Control Function
PDCP	Packet Data Convergence Protocol
PHY	Physical Layer
PNI-NPN	Public Network Integrated Non-Public Network
PLMN	Public Land Mobile Network ID
QoS	Quality of Service
RAN	Radio Access Network
RAPL	Running Average Power Limit
RB	Radio Bearer
RF	Radio Frequency
RLC	Radio Link Control
RRC	Radio Resource Control
RS	Radio Site
RU	Radio Unit
RX	Receiver
SBA	Service-Based Architecture
SCP	Service Communication Proxy
SDN	Software Defined Networking
SDR	Software Defined Radio
SMF	Session Management Function
SNPN	Standalone Non-Public Network
SNR	Signal-to-Noise Ratio
SoC	System on Chip
TAC	Tracking Area Code
TCP	Transmission Control Protocol
TX	Transmitter
UDM	Unified Data Management
UDP	User Datagram Protocol
UDR	User Data Repository
UE	User Equipmment
UP	User Plane
UPF	User Plane Function
URLLC	Ultra-Reliable Low-Latency Communication
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
vBS	Virtualized Base Station
VR	Virtual Reality
V-RAN	Virtualized Radio Access Network

Chapter 1

Introduction

1.1 Context

Fifth generation of cellular networks has posed new challenges in terms of performance and specific requirements. These demands encompass considerations such as reliability and quality of service, security and privacy, bandwidth availability, low-latency, scalability, and cost-effectiveness. Additionally, the advent of 5G has opened up new avenues for the deployment of private networks, more precisely Non-Public Networks (NPNs), making them a key interest of this research. Characterized as autonomous cellular networks with limited access, 5G NPNs represent a distinct facet of the evolving telecommunications landscape. Being autonomous means that they are not dependent on the telecom private service regulations. They are usually implemented for organizations, such as a campus or a factory, etc., based on the specific requirements of the organization. This gives ownership and control of the network to the organization.

This research is a component of the Route 25 (R25) initiative, a broader project aimed at shaping the future of mobility through the integration of proven technologies. The project's goal is to advance towards autonomous, intelligent, interoperable, and inclusive mobility solutions. Capgemini Engineering in Portugal is leading this research and innovation endeavor. The reason NPNs are being explored in the R25 project is to develop a potential infrastructure capable of supporting connected mobility services on managed highways, serving as an underlay to public cellular operators.

1.2 Motivation

This research is motivated by the imperative to evaluate the pre-existing 5G NPN network deployed for research purposes in the IT Porto Lab at FEUP. By analyzing its performance across various data rates, this study aims to assess key metrics including energy consumption, power consumption, and throughput. Additionally, the research aims to provide insights into the efficiency and effectiveness of the infrastructure, thereby contributing to advancements in 5G network optimization and deployment strategies.

The potential applications of non-public networks on the highways and in factories, ports, and airports of the future underscore the significance of this research. Understanding the performance nuances of the 5G NPN network is not only essential for optimizing its current applications but also lays the groundwork for future innovations and advancements in dedicated network technologies. This understanding holds particular significance for various industry verticals—distinct sectors or markets such as manufacturing, healthcare, or transportation—where tailored and reliable network solutions are instrumental for driving specialized applications and services. This work aspires to contribute valuable knowledge about the throughput limitations and causes of energy and power consumption in the technology that propels the capabilities of 5G NPNs, fostering a new era of cellular networks that is both robust and tailored for the needs of specific institutional environments.

1.3 Goals

As previously indicated, this study delves into crucial facets of 5G non-public networks, aiming to assess their performance across various data rate scenarios, specifically analyzing its impact on energy consumption. Before initiating the evaluation phase, it is important to understand network setup and be aware of all network's aspects. Based on the established testbed infrastructure of NPN at IT Porto, the evaluation process is made possible. There are three main objectives of this work. The first one is to validate the configurations of this testbed in order to achieve expected high throughput. Additionally, we aim to assess the energy consumption of disaggregated RAN components under various operational conditions. At last, we plan to evaluate both throughput and energy consumption simultaneously to better understand the network performance. This research aims to identify the factors influencing energy consumption, offering a comprehensive understanding of all usage influences. The findings from this research can identify components, configurations or implementation issues that impact runtime performance.

1.4 Document structure

Beside chapter 1 this document contains 4 more chapters. Chapter 2 explores fundamental topics and concepts needed for understanding the dissertation. It also presents state of the art and describes already known works related to the dissertation topic. In chapter 3, dissertation problem is defined and methodology that will be used to perform future work envisioned for this dissertation. In this chapter the already established setup is also explained. Chapter 4 details the measuring process and shows obtained results. Chapter 5 gives conclusion to the topics addressed in the document and suggests work to be done in future.

Chapter 2

Background and State of the Art

This chapter explains three key concepts needed to understand the topic of this dissertation. First topic is 5G and it's architecture, second is virtualized RAN, and third non-public private networks which are the biggest focus of the dissertation. Furthermore, in this chapter review of the previous works related to non-public networks and network performance in terms of energy consumption and throughput when using virtualized RANs is given. Four of the previous works are considered to hold the most significance and are referred to in the section 2.4.

2.1 5G

The fifth generation of cellular networks was developed in order to resolve the shortcomings of previous generations. With 5G the ability to reach peak rates in units of Gbps, while maintaining low latency and high reliability was made possible. Furthermore, harnessing the bandwidth capabilities of 5G enables the creation of robust connectivity, ensuring extensive coverage and seamless device integration within the ecosystem. This expands the usage of 5G even more [22].

The common classification of 5G communication services defines three categories. These categories are enhanced Mobile Broadband (eMBB), massive Machine-Type Communications (mMTC) and Ultra-Reliable Low-Latency Communications (URLLC) [11]. The first category focuses on improving the overall user experience, enabling high data rates, up to 20Gbps [15], and enhanced capacity. This category is well-suited for applications that demand high data speeds, such as ultra-high-definition video streaming, virtual reality (VR), and augmented reality (AR). The mMTC is designed to support a massive number of connected devices, up to 1 million connections per 1 GHz. They are particularly important in scenarios where a large number of devices need to communicate with the network simultaneously. This category is crucial for the Internet of Things (IoT) and applications with a vast array of connected sensors and devices, such as smart cities, industrial automation, and agricultural monitoring. The URLLC, as the name says, prioritizes extremely reliable and low-latency communication, latency of 1ms or lower. It is essential for applications such as autonomous vehicles or remote medical procedures [17].

In order to be able to fully understand the functionality of 5G networks we first must define their architecture. The three main components, shown in figure 2.1, that 5G consists of are [22]:

1. User Equipment (UE)
2. Core Network (CN)
3. Radio Access Network (RAN)

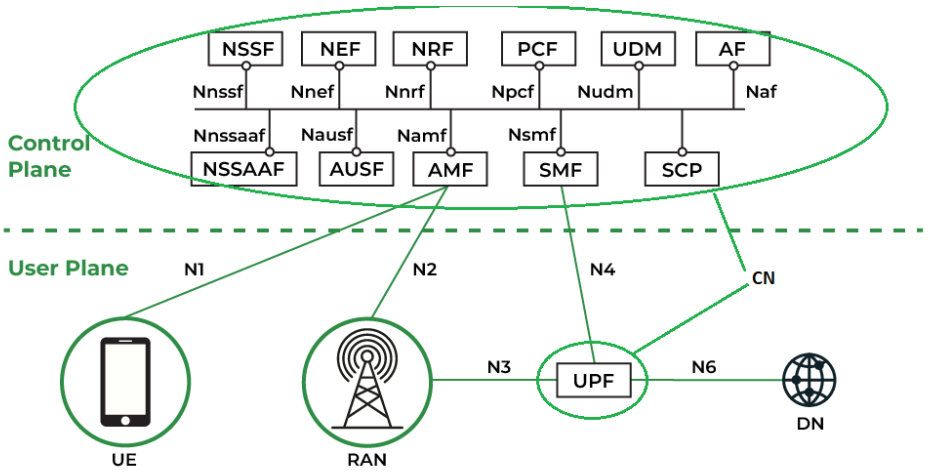


Figure 2.1: 5G architecture, figure based on [10]

2.1.1 User Equipment

User Equipment constitutes a diverse range of devices that are trying to access and utilize the 5G network. These devices include mobile phones, tablets, robots, cars, routers, and an array of Internet of Things (IoT) devices. UEs are equipped with specialized hardware components designed to facilitate seamless communication with both the Radio Access Network (RAN) and the Core Network [22]. This hardware enables UEs to engage in various communication modes, supporting not only traditional voice and data services but also catering to the specific requirements of emerging technologies such as autonomous vehicles, smart devices, and industrial IoT applications. UE represents the component of a telecommunications system positioned closest to the end user, serving as the primary interface where applications are executed and accessed.

2.1.2 Core Network

The Core Network in 5G networks is commonly known as the 5G Core (5GC). For security, the main functionality of the 5G Core is to ensure provision of requested services only to authenticated users, even when they are on the move. The CN needs to keep track of their movements, concretely UE movement, through Base Stations (BS) in order to maintain connectivity. The network function responsible for keeping track of the movements is called the Mobility Management Entity (MME).

The MME is a key component of the 5G mobile networks, which handles tasks such as tracking the location of UEs, managing mobility between different BSs (via handovers), and initiating paging procedures to locate UEs for incoming calls or messages. Thus, the MME plays a crucial role in ensuring seamless connectivity as UEs move within the network. What is also important is that cellular networks aim to guarantee a certain quality of service (QoS). This security, mobility and QoS is what differentiates cellular networks from Wi-Fi [22].

The 5G Core Network has a 3GPP Service-Based Architecture (SBA). SBA implies having a set of network functions (NFs) divided into smaller components called services. These services represent independent functions, running on the same underlying infrastructure, which can be used to create new network functions. Since we are working with multiple services, this means that multiple consumers can access service-based interface at the same time. Furthermore, based on the user demands, different amounts of resources are allocated to different users in order to provide scalability and improve performance [21]. Most common factors that influence resource allocation are traffic demand, response time and cost. For instance, in case of high traffic demand, some of the options are to provide more computing resources (CPU, memory etc.) to the UE, or assign more network functions to handle the traffic.

5G network architecture consists of User Plane (UP) and Control Plane (CP) shown before in figure 2.1. The User Plane consists of the UE, the RAN, the User Plane Function (UPF) and the Data Network (DN), while the rest of the functional elements are part of a Control Plane. The UPF and all of the functions in the Control Plane are part of the 5GC. The User Plane Function manages the transmission of traffic between the UE and the Data Network. Beyond forwarding IP packets, the UPF plays a pivotal role in policy enforcement, lawful interception, measurement of traffic usage, and enabling QoS. The UPF also takes charge of the tunneling process, encapsulating and decapsulating packets during their transit to and from the base stations. From the functions implemented in the Control Plane, two represent the highest significance [22]. These two are:

- AMF (Access and Mobility Management Function) - It is responsible for authorizing and giving access to the network, while also managing mobility of devices.
- SMF (Session Management Function) - It is responsible for managing sessions of each UE separately.

The other functions are:

- NSSF (Network Slicing Selection Function) - selects the best network slice available at the moment of the request by the user.
- NEF (Network Exposure Function) - exposed available services making them discoverable.
- NRF (Network Repository Function) - stores all of the 5G network functions (NFs).
- PCF (Policy Control Function) - develops and implements policies in 5G network.

- UDM (Unified Data Management) - generates user credentials and gives access to the authorized users.
- AF (Application Function) - allows applications to use network services.
- NSSAAF (Network Slice and Service Authorization and Accounting Function) - provides network slices and services authorization and accounting support.
- AUSF (Authentication Server Function) - authenticates user equipment.
- SCP (Service Communication Proxy) - manages and facilitates communication between different services within the network.
- BSF (Binding Support Function) - provides binding support function for authentication of user equipments.
- UDR (User Data Repository) - stores user related data.

2.1.2.1 Open source core network frameworks

There are a few frameworks that can be used to implement the core network. Most common ones lately are OpenAirInterface, Open5GS, free5GC, and Aether [23]. Some of the features of these open-source frameworks are shown in table 2.1.

Table 2.1: Open-source core network frameworks

	OpenAirInterface	Open5GS	free5GC	Aether
3GPP specification	Release 16	Release 17	Release 15	Release 15
Language	C	C++	Go	Go
Number of Contributors	Large and active	Large and Active	Medium sized	Medium sized
Tested RAN	Commercial, OAI O-RAN Simulators: UERANSIM, gNBsim	Commercial, OAI O-RAN Simulators: UERANSIM	Commercial Simulators: UERANSIM	Commercial, SD-RAN Simulators: UERANSIM, gNBsim

2.1.3 Radio Access Network

RAN comprises a set of base stations. Main function of base station is to provide network coverage in the desired area. These areas are further divided into smaller areas which are called cells. Each cell has at least one fixed-location transceiver called Base Transceiver Station (BTS) [11]. Base stations in 5G technology are known as gNB, where “g” stands for next Generation and NB for Node B. In general, BS is responsible for transfer of the packets between 5GC and sets of UEs. The transfer is possible after performing a set of functions. BS performs these functions which are

organized into five layers: 1.Radio-Frequency (RF), 2.Physical (PHY), 3.Medium Access Control (MAC), 4.Radio Link Control (RLC) and 5.Packet Data Convergence Protocol (PDCP) layer. This can be seen in figure 2.2. Key aspects of each layer are [22]:

- RF (Radio Frequency Layer) - RF layer manages digital processing, such as analog-to-digital conversion (ADC) and digital-to-analog conversion (DAC), as well as filtering and power amplification.
- PHY (Physical Layer) - This layer is responsible for implementation of Digital Signal Processing (DSP) functions, such as Fast Fourier Transform (FFT). It performs modulation/demodulation and coding/decoding functions.
- MAC (Media Access Control) - MAC controls which segments are sent when (scheduling). It handles buffering, multiplexing and demultiplexing of segments.
- RLC (Radio Link Control) - RLC handles segmentation and reassembly of data packets. It provides reliability in transmission and receivement of data segments by using error correction with automatic repeat request (ARQ).
- PDCP (Packet Data Convergence Protocol) - This protocol performs compression and decompression of Internet Protocol (IP) headers, ciphering/deciphering, integrity protection and verification and management of packet delivery.
- RRC (Radio Resource Control) - It is responsible for controlling the establishment, maintenance, and release of radio bearers (RBs) between the user equipment and the base station.

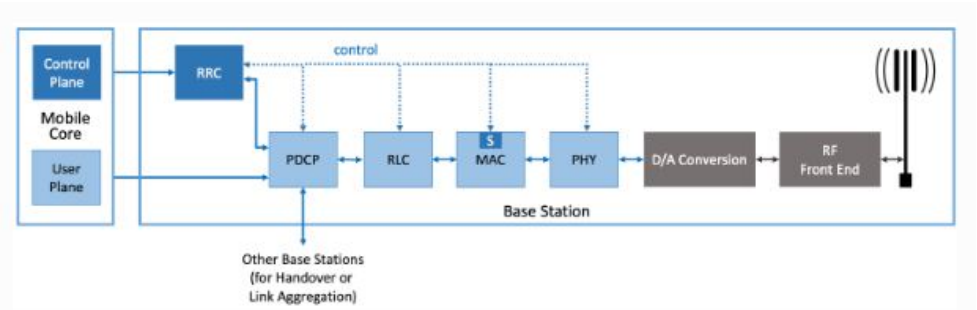


Figure 2.2: Base station layers, [22]

Executing functions within each layer requires a certain amount of energy. This necessitates the exploration of methods to decrease this energy consumption. The standard BS consists of remote Radio Unit (RU), transceiver, that connects to a BaseBand Unit (BBU). The RU is responsible for digital radio signal processing and the BBU processes multiple calls and forward's traffic, which is called baseband processing [15]. For virtualized RANs, some BS functionalities are moved to the cloud. This way energy efficiency is improved. Virtualized RANs are described in next section.

2.2 Virtualized Radio Access Networks (v-RANs)

Unlike the traditional RANs where functions are integrated in dedicated hardware, virtualized RANs have virtualized functions that can operate on generic hardware [15]. They are also called Cloud RANs. On the cloud, there are multiple virtual BBUs, which are deployed on a virtual machine or a container [5]. These RANs have the reduced number of BSs due to their baseband processing functions being centralized [11]. This kind of architecture allows for easy scaling of resources depending on the demand. Because of this, they are expected to be a good solution for reducing the energy consumption. Furthermore, they enhance network scalability, streamlines network management and maintenance, boosts spectral efficiency and network throughput.

In virtualized RAN, virtualized BBUs run on General Purpose Processors (GPPs). BBUs can be separated from their corresponding RUs and then pooled into a centralized BBU. Given that each pool of BBU has the capacity to cater to multiple RUs, virtualization becomes essential for establishing logically isolated networks. Every RU is connected via fronthaul link to its corresponding BBU pool, while BBU is connected through backhaul link to core network. This establishment of networks shares hardware resources such as CPU and memory among RUs. This approach creates the potential for dynamic allocation of hardware resources based on traffic demand, offering the prospect of substantial energy conservation.

Moreover, virtualized BBUs are further divided into two units, Distributed Unit (DU) and the Centralized Unit (CU) which can be seen in figure 2.3. This is called F1 interface and it is a standardized 3GPP interface in TS 38.470 [2]. For F1 interface, the DU processes the radio signal, and CU handles and oversees the RAN. From the figure 2.3 it can be seen that CU manages PDCP and RRC functionalities, while DU manages RLC, MAC, and PHY functionalities.

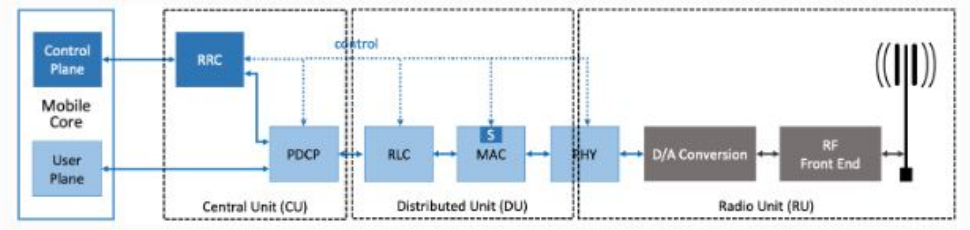


Figure 2.3: RAN split on CU, DU and RU, [22]

In terms of centralization, two types of v-RANs are recognized; fully centralized v-RAN, where all functions related to first three layers are located in BBU, and partially centralized v-RAN, where functions from layer 1 are transferred to the Radio Unit (RU). This can be seen in figure 2.4. The fully centralized type offers easier expansion of network coverage and improvement of network capacity, unlike the partially centralized RAN. However, partially centralized RAN requires low transmission bandwidth because the baseband processing is shifted from the BBU to the RU. Based on the application and user need, one or the other v-RAN is implemented.

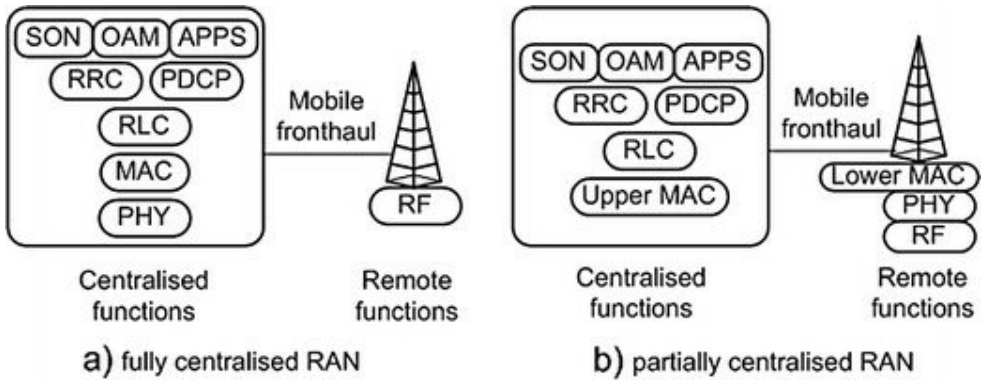


Figure 2.4: Fully and Partially centralized RAN, [9]

2.2.1 Open-source RAN frameworks

When it comes to open-source approaches related to virtualized RANs, there are constant developments due to the increasing involvement of researchers in its development. Some of the currently used open-source RAN platforms are srsRAN and OpenAirInterface (OpenAI or OAI) [5], but even the number of platforms is constantly increasing. Frameworks with their features are presented in table **??** [25].

Table 2.2: Open-source RAN frameworks

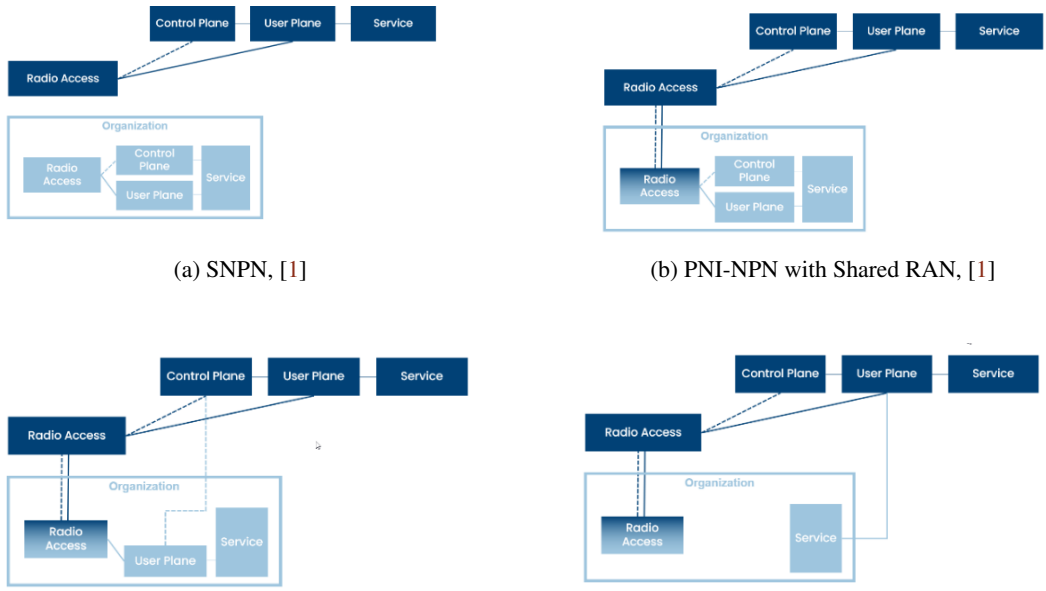
	OpenAirInterface	srsRAN	UERANSIM
Supported networks	4G-LTE and 5GNR	4G-LTE and 5GNR	5G-NR
Language	C	C++	C++
Supported RF	USRP B210, USRP X310, BladeRF, LimeSDR, EURECOM EXPRESSMIMO2 RF	LimeSDR, BladeRF, USRP X310	Only software based simulations
Community	Highly active	Active	Semi-active

2.3 Non-public networks (NPNs)

Non-Public Networks aspire to extend technologies, such as 5G, developed for public networks to private entities or network tenants. This is achieved by limiting network access exclusively to authorized terminals. There are two defined types of NPNs: standalone (SNPN) and Public Network Integrated (PNI-NPN) [6].

When it comes to NPNs, there are 4 distinguisable architectures, shown in figures 2.5a to 2.5d, recognized by 5G Alliance for Connected Industries and Automation (5G-ACIA) [1]:

1. SNPN is separated from the public network and all its functions are located inside the organization .
2. PNI-NPN where part of the RAN is shared among the owner and the mobile operator. Network functions are separated from the public network and located inside the organization.
3. PNI-NPN with shared RAN and control. Public network performs the control tasks, but the NPN is still located inside the organization. Here, the network slice or Access Point Name (APN) can be dedicated to the owner.
4. NPN is hosted by a public network. NPN is completely part of the public network, end-to-end, and uses it for all purposes. Network slice is end-to-end.



(a) SNPN, [1]

(b) PNI-NPN with Shared RAN, [1]

(c) PNI-NPN with Shared RAN and Control Plane, [1]

(d) NPN hosted by the public network, [1]

Figure 2.5: Different types of Non-Public Networks

The SNPN is identified by a combination of Public Land Mobile Network ID (PLMN ID) and Network IDentifier (NID). This combined ID is broadcast by the Radio Access Network infrastructure, allowing authorized User Equipments to discover the SNPN. Once discovered, the secured attachment procedure begins, distinguishing it from a regular PLMN. To connect to the SNPN and access its features, an UE must be enrolled using a dedicated subscriber list. UEs registered in an SNPN can also access PLMN services, if needed, by registering again with a PLMN using the SNPN User Plane. In this case, the SNPN acts as an untrusted non-3GPP access network. The reverse scenario is also possible, where UEs on a PLMN can access SNPN services while connected to the PLMN. SNPN is a good choice for organizations that need higher-level control over mobile network and customized configuration.

On the contrary to SNPNS, PNI-NPNs integrate with PLMN to a certain level. The sub-public network within this kind of NPN supports functions by PLMN, while sub-private network's functions are isolated from PLMN. Only two ways support provision of PNI-NPNS:

- Implementation of Data Network Names (DNNs)
- Implementation of a network slice

In the first case, the PLMN establishes a mobile pipeline to direct NPN traffic to a dedicated mobile gateway. This gateway is responsible for routing traffic to a local area data network (LADN). In the second case, the PLMN allocates specific resources to create a dedicated slice exclusively for non-public services.

2.4 State of the Art

[25] explores development and implementation of 5G Non-Public Network (NPN) by leveraging the open-source software established at IT Porto lab, at FEUP campus environment. It details the entire process of establishing an effective setup for conducting required measurements and provides guidance on the utilization of the OpenAirInterface (OpenAI or OAI) framework. The measurements considered in this research are focused mainly on throughput and latency. In a number of papers researchers have not been able to obtain the expected results when evaluating the network performance, which was also the case in this dissertation work. The throughput was found to be lower than what has been reported by the scientific community, and the latency from the User Equipment (UE) to the Core Network (CN) was excessively high. The observed throughput was thought to be a result of imperfections in the physical infrastructure. It was believed that GPRS Tunneling Protocol (GTP) tunnels may contribute to latency. Both of this aspects require further exploration.

As for the virtualized Radio Access Networks (v-RANs), [16] delved into the groundbreaking concept of virtualizing radio access networks within the implementation of 5G technology. Significant advancements in techniques like Network Function Virtualization (NFV) and Software Defined Networking (SDN) have been introduced for the virtualization of processing, departing from the traditional BBU processing method. NFV and SDN enable the dynamic allocation of computational resources and the adjustment of power consumption based on traffic loads in 5G networks, which is why these techniques were considered. In this study, various functional splits outlined by European Telecommunications Standards Institute (ETSI) were examined for the dual-site architecture. The initial layer encompassed the central cloud site (CS) responsible for processing the second layer of the edge cloud, known as the remote sites (RS). The third layer, comprising the radio units, was tasked with providing coverage for user devices. According to the percentage of active radio heads in the network, the power consumption was evaluated. It was found that the power consumption varies in the range of $1.7x10^{12}$ to $4x10^{12}$ Watt for 100% and 40% usage respectively. In general, the primary objective of the paper was to analyze the trade-off between mid-haul bandwidth requirements and the total power consumption based on different functional

splits, aiding operators in optimizing parameters in real networks to achieve high performance while maintaining energy efficiency.

[4] concentrated on addressing the less-explored issue of uplink power consumption since it poses greater challenges in characterization compared to the downlink due to its non-linear relation between load and resource usage. For this work open-source platform srsLTE was deployed. Initially, they quantified power as a function of Signal-to-Noise Ratio (SNR) and airtime. Results revealed that the power increased with the increase of SNR. This was attributed to the utilization of higher Modulation and Coding Schemes (MCSs) on more favorable channels, inducing greater computational load. This helped identify a trade-off between the power consumption of the User Equipment (UE) and the virtual Base Station (vBS). While the UE can conserve energy by reducing transmission power, this action exacerbates SNR, leading to increased vBS power consumption. It was also noted that for a specific target throughput, various configurations with different power consumption can be identified. In scenarios with favorable channel quality, configurations featuring higher MCS and lower airtime prove effective in reducing power consumption. Conversely, for lower SNR values, it is more energy-efficient to use lower MCSs and higher airtime.

In paper [3], a few open source softwares were evaluated and compared in terms of performance. For the RAN they have chosen to compare srsRAN and OAI frameworks, while for the core they compared Open5GS and OAI software. Also, as the Universal Software Radio Peripheral (USRP), USRP B210 was used. Among the four open-source software evaluated, srsRAN Project and Open5GS were found to be the most user-friendly due to their straightforward configuration files, while OAI RAN and OAI 5GC were more challenging to deploy. Furthermore srsRAN Project and Open5GS showed more robustness and were not easily disturbed by the errors. It was possible for the UE to re-establish the connection in case of loss, for which the OAI required to be restarted. In comparing srsRAN and OAI RAN with OAI 5GC, srsRAN offers superior UL rates and OAI RAN delivers better DL rates and more stable latency. Between Open5GS and OAI 5GC using srsRAN, no significant difference was noted in UL and DL rates or latency. Overall, srsRAN and Open5GS are more user-friendly and robust, while OAI RAN excels in DL rates and latency stability. The conclusion was that the functional split might be a next step in the development.

The studies highlight important aspects of 5G network implementation with open-source software, but several questions remain. The reasons for lower-than-expected throughput and high latency observed in [25] need further exploration, especially regarding infrastructure imperfections. While [16] analyzes power consumption trade-offs in virtualized RANs, the optimal functional splits for various network conditions still need investigation. [4] points out the trade-off between UE and vBS power consumption, but further research is needed to find the best configurations for different SNR values. Lastly, [3] suggests that functional splits in srsRAN and OAI frameworks could be beneficial, but this requires more thorough examination. All of this is stated in the table 2.3.

Table 2.3: Future work regarding state of the art

Related work	Left to explore
Open source non-public 5G networking in licensed and unlicensed bands [25]	Reasons for lower-than-expected throughput and high latency
Optimum functional splits for optimizing energy consumption in v-RAN [16]	Optimal functional splits for different network conditions
Experimental evaluation of power consumption in virtualized base stations [4]	Find the best configuration to achieve good SNR
A comparative analysis of open-source software in an E2E 5G standalone platform [3]	Performance of OAI framework for RAN functional split

This study will address the throughput limitations encountered in previous research, as well as this one, and will thoroughly evaluate the topic of energy consumption. It will also assess the network performance of the OAI framework for a split RAN, as posed by earlier research.

Chapter 3

Methodology

This chapter explain the problems addressed in the research and details the experimental setup, data collection procedures and techniques employed to investigate these issues.

3.1 Problem formulation

Open-source 5G networks are still an emerging concept, undergoing continuous development and evolution. Due to this, the documentation currently accessible is not extensive, posing a challenge for research in this area.

With the testbed developed at IT Porto that uses open-source software framework OpenAI (OAI), few performance issues were identified in the previous works. This paved the way for addressing the main problems in this research. One of the problems this research addresses is the maximum achievable throughput in the network and understanding its limitations. The second problem involves examining the total energy consumption of the RAN components and the processor, and understanding their dependency on throughput. Lastly, the research includes a deep examination of the processor's energy consumption by monitoring CPU performance counters.

By utilizing the OpenAI infrastructure, the O-RAN Alliance Scientific Community registered the maximum throughput of 200 Mbps which is a value 4 times larger than the one obtained at the IT Porto lab in the previous research [25]. For this reason, this research aims to first identify the existing limitations of our network's throughput and then, if possible, improve the network in order to achieve the maximum throughput reported in other studies.

Moreover, in the previous research and for the same testbed, it was noticed that there was an unbalanced energy distribution among Radio Access Network (RAN) components (CU, DU and UE), which is another problem that needs to be further investigated. This involves identifying the factors contributing to energy consumption and understanding what to monitor. In our case, we are focusing on understanding this imbalance in energy consumption, and further investigating how does energy consumption vary due to the changes in throughput.

A significant aspect of this investigation is attributing energy consumption to the processor and analyzing how it varies with changes in throughput. Understanding the processor's power

consumption is crucial, as it often represents a significant portion of the overall energy usage. Additionally, this knowledge is essential for the real-time management of virtualized RANs. Knowing the energy consumed by processors enables better resource allocation and load balancing, ensuring that the network operates efficiently without compromising performance.

By examining these elements, a comprehensive understanding of both throughput limitations and energy efficiency in the testbed environment will be provided.

3.2 Experimental setup

The implemented research network is a 5G NPN Standalone network. The Core Network was implemented using Open5GS for its characteristics of supporting the needed number of network functions, having a constantly updated framework and being compatible with more recent release of 3GPP. The chosen framework for the RAN was OpenAirInterface (OAI) because it supports CU and DU split and is compatible for various RFs, such as Universal Software Radio Peripherals (USRPs), which is a needed requirement in our research.

To set up the OAI RAN infrastructure, the OAI software was initially installed on a machine running Ubuntu 22.04. Since the RAN design adopts a F1 split scheme, it is divided into the Centralized Unit (CU) and the Distributed Unit (DU). Consequently, the experimental setup involved a testbed comprising four machines: one for the Core Network (CN), one dedicated to the DU, one managing the CU, and one for the User Equipment (UE). To enable communication between the CU and DU machines, direct connectivity was established by using an Ethernet cable connected to the network interface, with IP addresses manually configured, [25].

DU and UE are each connected to one USRP, which we called USRP DU and USRP UE, via USB 3.0. USRP devices are software-defined radios (SDRs) used for RF applications, offering frequency coverage from approximately 70 MHz to 6 GHz [24]. Designed for cost-effective experimentation, they support the USRP Hardware Driver (UHD) software, enabling seamless code reuse from existing designs and compatibility with open-source applications. They support number of applications, such as FM and TV broadcast, cellular, GPS, WiFi, and number of frameworks such as RF Network-on-Chip (RFNoC), GNU Radio, HDL Coder, and MathWorks MATLAB and Simulink software [15], which is why they are also used in out setup. In our testbed, the two USRPs communicate between themselves through a coax cable where one's receiver (RX) input is connected to the other's transmitter (TX) input. The testbed also includes a connection to a 4-channel power meter, specifically the Chroma digital power meter model 66204. Initially, two channels of the power meter were connected to each USRP, while the remaining two were connected to the CU and DU machines. Later, we only used used 3-channels since the focus was on CU, DU and UE. The whole testbed can be seen in figures 3.1, 3.2 and 3.3.

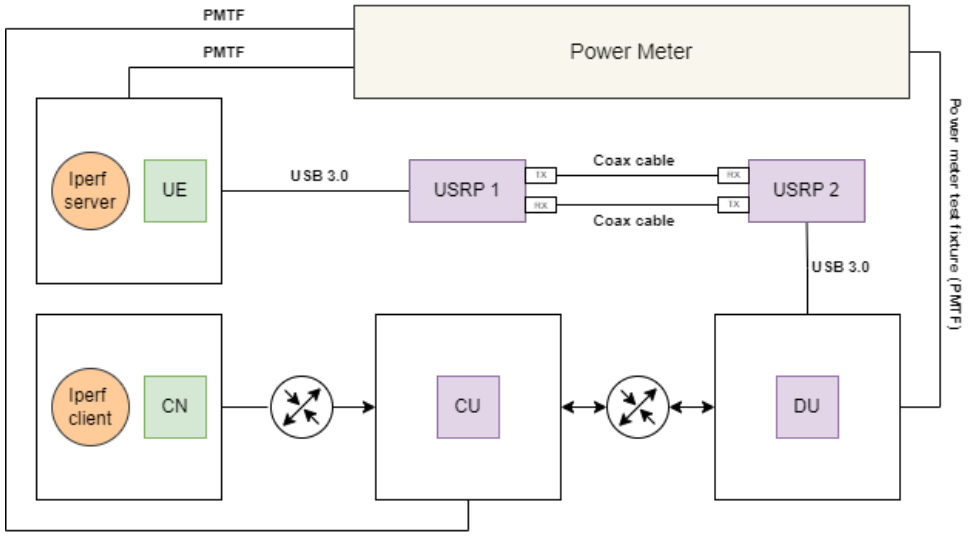


Figure 3.1: Testbed

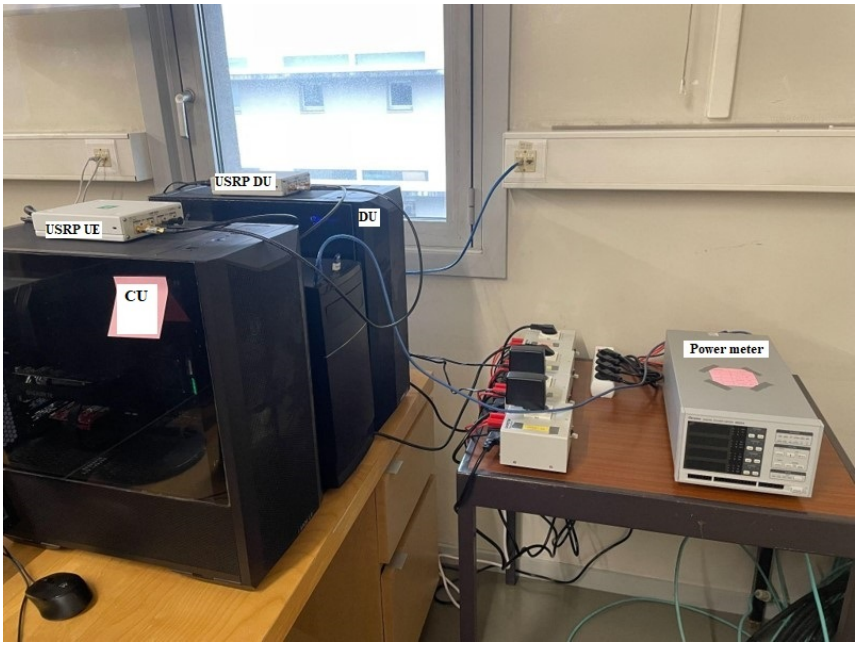


Figure 3.2: Testbed in the lab, CU, DU and USRPs

Furthermore, considering physical layer parameters, there are a few that are important for the functionality of the network. Frequency band is n78(TDD, 3300MHz - 3800MHz) while the subcarrier spacing is 30kHz. The carrier bandwidth covers 106 Resource Blocks (RBs), which corresponds to 38.16 MHz. This provides a wide spectrum for communication. Additionally, the carrier frequency is set at 3619.2 MHz, chosen to optimize wireless connectivity within the designated frequency band. All of this can be consulted in table 3.1.

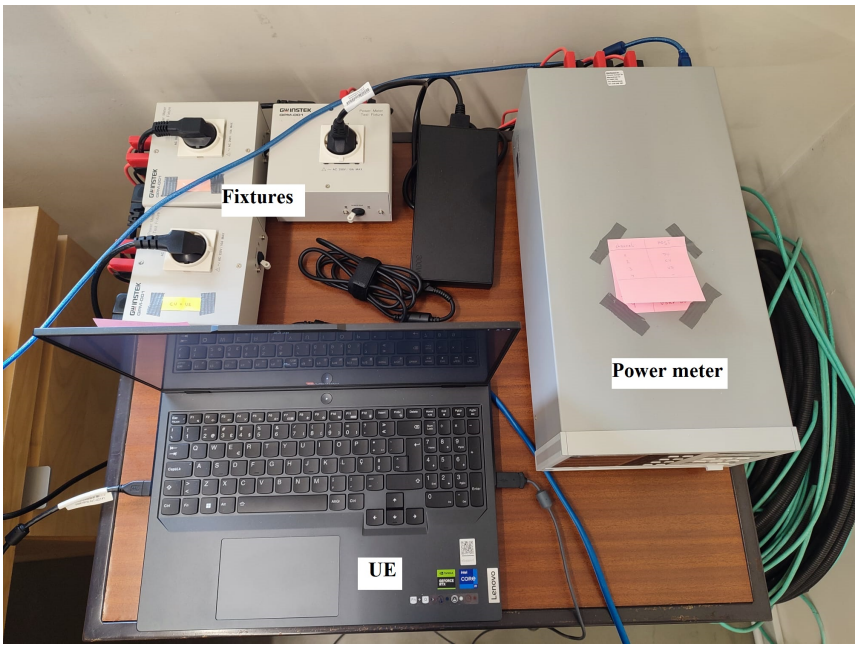


Figure 3.3: Testbed in the lab, UE and power meter

Table 3.1: Physical layer parameter configuration

Physical layer parameters	Configuration
Frequency band	n78 (TDD, 3300MHz - 3800MHz)
Subcarrier spacing	30kHz
Resource blocks	106
Carrier bandwidth	38.16 MHz
Carrier frequency	3619.2 MHz

3.2.1 Hardware specifications

Table 3.2 details the hardware specifications of all components in the system, [25, 19].

3.3 Open-source 5G NPN Deployment

After setting up the testbed, explained in section 3.2, the network had to be deployed and validated. This section details the deployment of the network using OAI framework and explains how to initialize the network components.

3.3.1 Virtualized RAN deployment

RAN was deployed on an Ubuntu 22.04 machine using OAI which is compatible with Open5GS chosen for the core network [25]. Steps to take in order to set up RAN are:

Table 3.2: Hardware specification

Devices	Specifications
Core	Dell PowerEdge R710 server machine
RAN	CPU: 1x Intel Core i9-13900K 24-Core (2.2GHz-5.8GHz) Disk: 1TB HDD RAM: 32 GB Motherboard: ASUS TUF Z790-Plus Gaming D4 GPU: Gigabyte GeForce® GTX 1650 D6 OC Rev.2 4G Network: 2x WL-PCI Express TP-Link TX401	10GB PCI Ethernet Cooling: 1x Cooler CPU Noctua NH-D15 chromax.black
UE	CPU: 1x Intel Core i9-14900HX 24-Core (2.2GHz-5.8GHz) Disk: 2x M.2 SSD up to 1TB RAM: 32 GB Motherboard: ASUS TUF Z790-Plus Gaming D4 GPU: NVIDIA® GeForce RTX 4070 8GB Network: Gigabit Ethernet, 1x RJ-45, supports Wake-on-LAN | Wi-Fi® 6E, 802.11ax 2x2 Wi-Fi® + Bluetooth® 5.1 or above
RUs	USRP B210

1. **Cloning the OAI repository from a gitlab repository to a specified repository on our machines:**
   Since this work is a part of a larger project, each individual had to use their specified folders on all machines to avoid interfering with others' work. Also, the version **2023w45** was the one being used. After navigating to the designed folders, the repository can be cloned:

   ```
   git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
   ```

2. **Building the OAI and all needed packages and simulators:**
   Navigating to openairinterface5g/cmake-targets/ folder and executing the command to:

- build and compile OAI,
- build external packages (-I),
- create new radio softmodem (–gNB)
- create New Radio (NR) user equipment softmodem (–nrUE),
- add USRP support (-w USRP)

```
./build_oai -I --gNB --nrUE -w USRP
```

3. **Establishing connections between network components and updating their configuration files:**

The connection between the CU and DU components via Ethernet was established by updating the *local_s_if_name* parameter in the *cu_gnb.conf* and *du_gnb.conf* files, appendix A, setting it to the enp7s0 interface. Additionally, IP addresses were configured manually to ensure these components are part of the same network and can communicate effectively. The assigned IP addresses were as follows:
CU's IP address: 10.42.0.1
DU's IP address: 10.42.0.2
These settings were implemented by adjusting the *local_s_address* (referring to the local device's IP address in *cu_gnb.conf* and *du_gnb.conf*) and *remote_s_address* (referring to the remote device's IP address in *cu_gnb.conf* and *du_gnb.conf*).

As for the USRPs (RUs), initially one USRP was connected to the CU, as the the UE was also on the same device, while the other USRP was connected to the DU device via USB. After separating the CU and UE, the USRP connection was moved to the UE.

3.3.2 Core Network and UE deployment

Core Network was deployed using Open5GS for the reasons explained in 2.1.2.1. To utilize the UE, the core network must register and authenticate it. This requires that the core network has access to the configuration information stored in the UE's *ue.conf* file:

```
uicc0 = {
    imsi = "208930100001124";
    key = "465B5CE8B199B49FAA5F0A2EE238A6BC";
    opc= "E8ED289DEBA952E4283B54E88E6183CA";
    dnn= "internet";
    nssai_sst=1;
    imeisv="6754567890123413"
}
```

The critical information includes the International Mobile Subscriber Identity (IMSI), Subscriber Key (K), and Operator Key (OPc). Updating the Public Land Mobile Network (PLMN) values, specifically the Mobile Country Code (MCC), Mobile Network Code (MNC), and the Tracking Area Code (TAC) on the core, is essential. In our scenario, the MCC is 208, the MNC is 93, and the TAC is 7. These values need to align with the corresponding values extracted from the UE's IMSI.

3.3.3 Network initialization and validation

To ensure that all components have the correct dependencies available for the communication, certain steps following a specific order need to be performed when initializing the network. These steps are:

- 1. Start the Core

- 2. Start the CU
- 3. Start the DU
- 4. Start the UE

Starting the Core first provides the overall network functionality. CU acts as operations' control unit, while DU provides communication with UE, which is why it cannot be initiated after the UE. If the network was successfully launched, terminal will show confirmation messages about the established connection between components. It is important to take notice about the USRPs being detected as well.

3.3.3.1 CU setting up and initialization

To set up the CU, the initial step involves configuring the *cu_gnb.conf* file, which defines the CU's parameters and specifications. Here are the essential configurations:

- Specify the TAC and PLMN (MCC/MNC) values at the beginning of the file to match the network setup. As previously said, MCC is 208, MNC is 93, and the TAC is 7.
- Set local_s_if_name to enp7s0, which identifies the interface connecting the CU and the DU. Assign local_s_address and remote_s_address with the CU's and DU's IP addresses, respectively.

For the AMF parameters:

- Configure amf_ip_address to 10.0.0.2, which is the Core's IP address.
- Set the network interface to enp8s0 with IP 10.0.0.3, connecting the gNB to the Core.

To initialize the CU, execute the following command, which includes the necessary components for the startup process:

```
sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/
CONF/cu_gnb.conf --sa -E --continuous-tx
```

This command selects the 5G gNodeB binary (nr-softmodem), specifies the configuration file (cu_gnb.conf), runs in standalone mode (–sa), enables the 3/4 sampling mode (-E) to accommodate the B210 USRP's requirements [25], and enables continuous transmission (–continuous-tx).

3.3.3.2 DU setting up and initialization

To set up the DU, the initial step involves configuring the *du_gnb.conf* file, which defines the DU's parameters and specifications. Here are the essential configurations:

- Specify the TAC and PLMN (MCC/MNC) values at the beginning of the file to match the network setup.

- Set local_s_if_name to enp7s0, the interface name connecting the CU and the DU. Also, configure local_s_address and remote_s_address with the CU's and DU's IP addresses, respectively.

To initialize the DU, execute the following command, which includes the necessary components for the startup process:

```
sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/
CONF/du_gnb.conf --sa -E --continuous-tx --T_stdout 2 --T_nowait -q
```

This command selects the 5G gNodeB binary (nr-softmodem), specifies the configuration file (du_gnb.conf), runs in standalone mode (–sa), and enables the 3/4 sampling mode (-E) to accommodate the B210 USRP's requirements. It also enables continuous transmission (–continuous-tx) and prints the output log in a readable way (–T_stdout 2 –T_nowait -q).

3.3.3.3 UE initialization

To initialize the UE execute the following command: To initiate the UE, the complete command would be:

```
sudo ./nr-uesoftmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/
CONF/ue.conf -r 106 --band 78 -C 3619200000 --ue-fo-compensation
--sa -E --ue-rxgain 80
```

This command selects the 5G User Equipment binary (nr-uesoftmodem), specifies the configuration file (ue.conf), runs in standalone mode (–sa), and enables the 3/4 sampling mode (-E) to accommodate the B210 USRP's requirements. It also sets the bandwidth in terms of resource blocks (-r 106), sets the frequency band (-band 78), sets downlink carrier frequency in Hz (-C 3619200000) and enables frequency offset compensation (-ue-fo-compensation).

3.4 Experimental method

Previous three sections have explained the problems at hand, experimental setup and how to deploy the network. This section details the methodology employed for conducting measurements, encompassing the use of diverse tools, and analysis of the collected data to obtain valuable insights.

By using the power meter, at first the power consumption of the CU, DU and two USRPs was measured, while later on we focused only on the CU, DU and UE. This device lays out the total power consumption of the device it is connected to. We chose to record all channels at the same time for the number of parameters, but primarily focusing on the evolution of instantaneous power consumption in watts (W).

3.4.1 Power meter configuration

To configure the power meter, the Windows Operating System (OS) was installed on another machine, where the power meter software, known as SoftPanel, was deployed. Through the SoftPanel software, channel configuration was set to Integration mode. The starting window of the SoftPanel tool can be seen in figures 3.4.

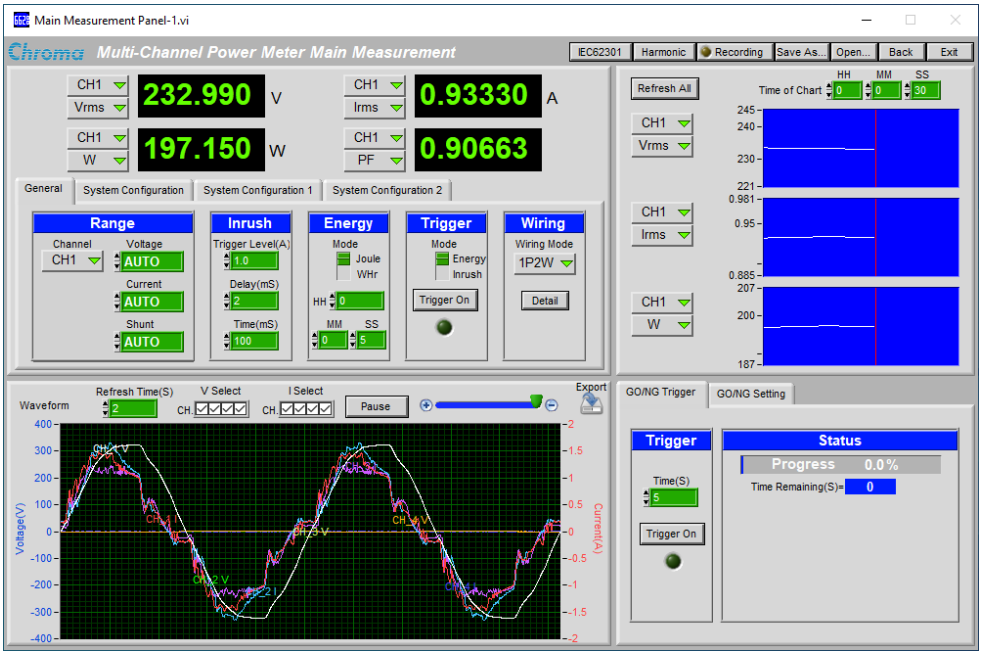


Figure 3.4: SoftPanel software tool

Figure 3.5 shows available operating modes for the power meter: average, windows and integration mode. Integration mode, compared to the others, allows for the real-time power monitoring, [7], which is the main reason we chose to follow this measurement path. It also provides more accuracy since the power is calculated by integrating the instantaneous power over user defined period of time (T_i), unlike the average and window modes where time periods are much smaller (T_1, T_2,...; T_w). By this it is ensured that short-term fluctuations and variations in power consumption are accounted for, resulting in a precise measurement of the total energy used.

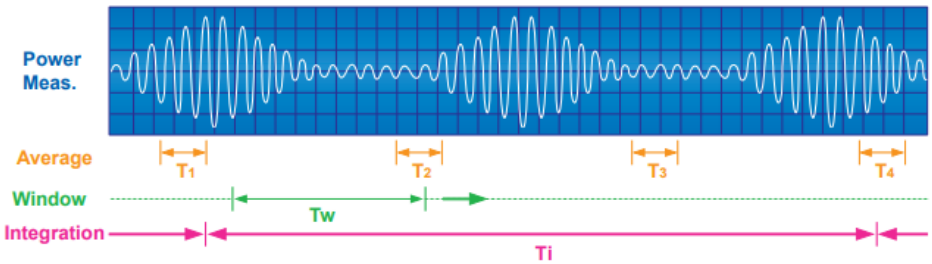


Figure 3.5: Average, window and integration operating modes for the power meter, [8]

3.4.2 Generating data with *iperf*

In addition to just measuring the power consumption with the power meter, we wanted to measure power consumption while simultaneously generating different amounts of data. To introduce traffic and monitor the obtainable throughput *iperf3* tool was used. *Iperf* tool is used for active measurements of the maximum achievable bandwidth that reports also the packet loss, [13]. We were performing measurements for downlink traffic. To do so, the UE performed as a server and CN as a client. The commands for introducing traffic are:

At the server side:

```
iperf3 -s -B <IP address>
```

At the client side:

```
iperf3 -t <time in seconds> -c <IP address>
```

The breakdown of the commands is following, [14]:

- **-s**: tells *iperf3* to run in server mode. In this mode, *iperf3* listens for incoming connections from *iperf3* clients to perform bandwidth tests.
- **-B <IP address>**: specifies the IP address to which the *iperf3* server should bind. Since we have only one server we would set the UE's IP address here.
- **-t** <time in seconds>: sets the transmission time.
- **-c <IP address>**: tells *iperf3* to run in client mode and connect to the *iperf3* server at the specified IP address (UE).

For UDP traffic, the command on the client side had to be slightly changed:

```
iperf3 -t <time in seconds> -c <IP address> -u -b <number of b/s>
```

In this case, the added flags stand for, [14]:

- **-u**: uses UDP traffic.
- **-b <number of b/s>**: sets the target bandwidth to n bits/sec. We used 4 different bandwidths, 8, 16, 32 and 64M bits/sec.

3.4.3 Measuring power consumption with RAPL

Apart from just looking into the power consumption of the whole device, it was interesting to understand the portion of power consumption attributable to the processor. RAPL, which stands for Running Average Power Limit, is an Intel processors' tool used to report the accumulated energy consumption of various system-on-chip (SoC) power domains [12]. This allows the developers to tune a system's performance based on the power usage. There are usually four domains:

- Package domain - measures the energy consumption of the whole processor package, including all cores and non-core components such as the integrated memory controller and cache.
- Power Plane 0 (PP0) - measures energy consumption of all cores of the processor
- Power Plane 1 (PP1) - measures energy consumption of processor graphics
- DRAM - measures energy consumption of Random Access Memory attached to the memory controller

In our case, we used this tool to measure the energy consumption of the processor in Package domain. For these measurements, we used a tool called *turbostat*, which can report on processor topology, frequency, idle power-state statistics, temperature, and power. The command to measure the power is:

```
sudo turbostat --out <output_file> --cpu package --quiet --debug -i
<seconds> --num_iterations <number>
```

The breakdown of the command is:

- **–out <output_file>**: saves the output to a file. We used .csv file to save the output.
- **– cpu package**: specifies that the statistics should be taken for CPU package.
- **–quiet**: minimizes the unnecessary information.
- **–debug**: displays additional information about the system configuration.
- **-i <seconds>**: sets the interval time length. Default is 0.5s, we used 1s.
- **–num_iterations <number>**: sets number of iterations. We used 300 iterations which corresponds to 5 minutes.

3.4.4 Monitoring CPU events with *perf*

To further elaborate on the issue, we utilized the *perf* tool to examine cache misses and cache references, aiming to clarify the observed power consumption. Concretely, the intention was to observe the potential co-occurrence of these cache events and high power consumption. *Perf* is a Linux profiling tool based on perf_events interface that uses CPU performance counters [28]. CPU performance counters are hardware registers that count hardware events such as number of executed instructions and cache misses. By using this tool, we aimed to identify which functions contribute the most to power consumption. Furthermore, we have taken notice how the number of cycles and cache miss ratio changes for different data rates and for the functions that were considered as the most relevant.

To record cache events, the most used commands are *perf record* and *perf report*. As the name says, *perf record* is a command used to record data:

```
sudo perf record -e cycles/period=100000/,r4F2E/period=10000/,
r412E/period=10000/:P -b -p <PID> -g --all-user -o <output_file>
-- sleep <time_in_seconds>
```

The explanation of the command:

- **-e:** is an event selector, the following cycles/period=100000/,r4F2E/ period=10000/,r412E/ period=10000/:P are the events we recorded. r4F2E stands for cache references and r412E for cache misses.
- **-p <PID>:** records events on existing process ID.
- **–all-user:** configures all events to run in user space.
- **-o <output_file:** store output in output file.
- **– sleep <time_in_seconds>:** specifies the time interval for which the perf will run.

The output file of this command is .data binary file. Therefore, the output file needs to be parsed to extract important information. For us, as already mentioned, cycles/period, cache misses and cache references held the highest importance. In order to parse the files we used the *perf report* command which reads the .data file:

```
sudo perf report --no-inline --stdio -n -s symbol -t, --children
--call-graph=none -F +overhead,overhead_sys,overhead_us,
overhead_children,sample,period -i <record_output_file>.data >
<output_file>
```

In this command:

- **–no-inline:** disables inlining in functions.
- **–stdio:** defines standard output.
- **-s symbol:** sorts symbols by name.
- **– children:** includes children processes in the output.
- **–call-graph=none:** disables recording of a call graph.
- **-F +overhead,overhead_sys,overhead_ us,overhead_children,sample,period:** specifies field to include in the output file.
- **-i <record_output_file>.data > <output_file>:** specifies the input file and redirects it to an output file. Output file was in the format .txt.

Chapter 4

Results Analysis

This chapter details the results obtained from the performed measurements and explains their evaluation process.

4.1 Defining baselines - Initial measurements

When it comes to the energy consumption analysis, it is essential to understand the behaviour of the devices in terms of power and energy consumption. The analysis provides insight into the efficiency of used resources and if they can be dynamically adjusted based on demand. This ensures optimal operational efficiency and sustainability in technological applications.

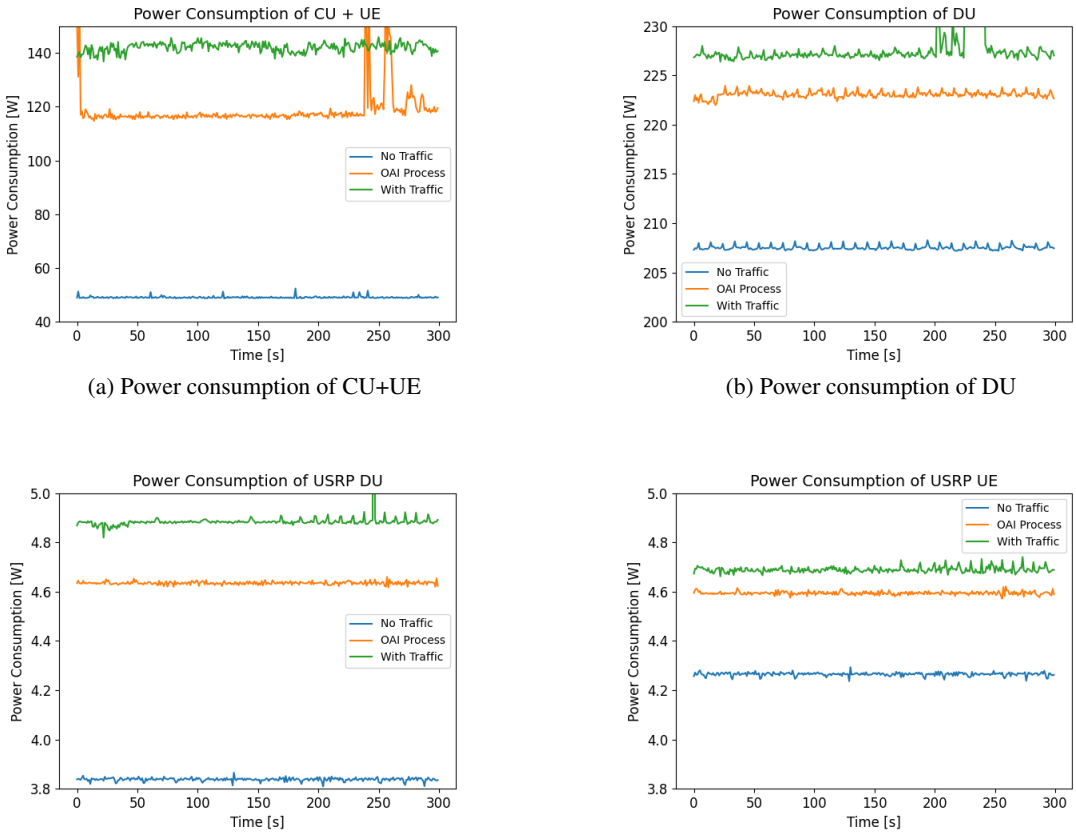
The relevant devices for this part of the analysis were two USRPs, CU, DU and UE, while in further steps we will only examine CU, DU and UE. By connecting these devices to the power meter, via power meter test fixtures, the total power consumption was measured. Three cases were evaluated:

1. Baseline case - Neither OAI process nor traffic was introduced for this measurement.
2. OAI process started - The OAI process was running on all devices.
3. Traffic introduced - In addition to the OAI process running, TCP traffic was also introduced.

Evaluating these three cases gave insight into the ranges of power consumption and set a baseline for future measurements.

4.1.1 Impact of core frequency scaling configuration

Measurements were taken for the time of 5 minutes and the results of measurements can be seen in figure 4.1. Initially we had both CU and UE deployed on the same device. Based on these results a significant difference in the power consumption of the CU + UE device and DU device was noticed. The DU consumed nearly twice as much power, which was an unexpected outcome since both devices have the same hardware components. Additional measurements confirmed this behavior. This lead to further investigation of the hardware specifications in the BIOS settings.



(a) Power consumption of CU+UE

(b) Power consumption of DU

(c) Power consumption of DU USRP

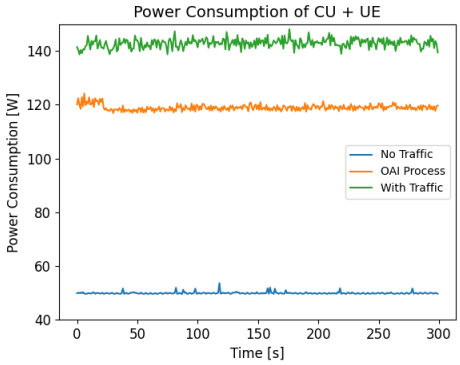
(d) Power consumption of UE USRP

Figure 4.1: Power consumption of the devices - Initial measurement

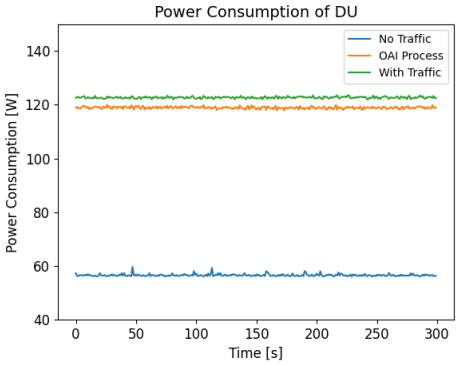
Looking into the BIOS settings lead to an interesting conclusion. It was noticed that the option **Core Ratio Sync** was differently configured in the CU and DU. For the CU it was set to *Auto*, while for the DU to *All Cores*. *All Cores* is a rather useful feature because it maximizes the performance of the processor by setting all cores to operate at the same frequency, but it also causes more power consumption. On the other hand, *Auto* implies having individual frequency scaling based on workload which ensures dynamic power management and less power consumption. Understanding these two options explained the difference in the power consumption of the machines. By setting **Core Ratio Sync** to *Auto* on the DU, the issue of the high power consumption was solved. The measurements were performed again. The resulting plots can be seen in figure 4.2. In all future measurements, **Core Ratio Sync** option was set to *Auto* on both CU and DU.

4.1.2 Obtaining baselines

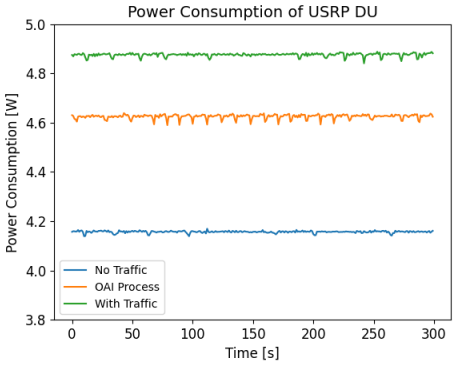
Shortly after resolving the issue of high power consumption on the DU device, the need to separate the CU and UE devices arose. This necessitated installing OAI dependencies on the new device and connecting it to one of the USRPs. New measurements were taken. As our objective was to measure the power consumption of the UE alongside the DU and CU, we disconnected the USRPs



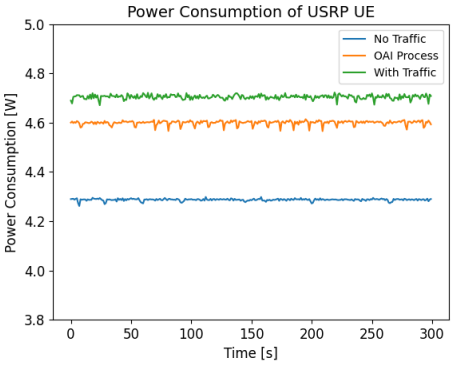
(a) Power consumption of CU+UE



(b) Power consumption of DU



(c) Power consumption of DU USRP



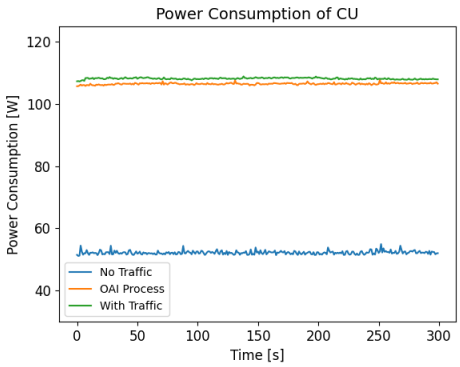
(d) Power consumption of UE USRP

Figure 4.2: Power consumption of the devices - After solving DU power consumption issue

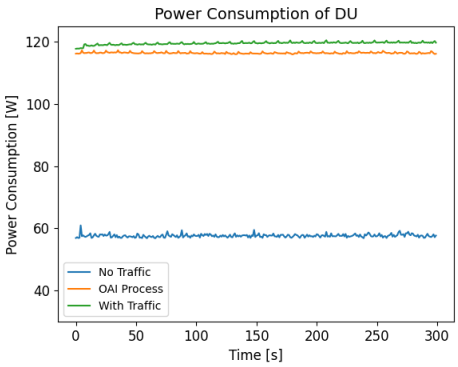
from the fixtures. Subsequently, all measurements in the further work focused exclusively on the three devices: DU, CU, and UE. The measurements were taken for the time of 5 minutes. The results of the measurements show that DU consumes the most power even without traffic which can be seen in figure 4.3. Moreover, CU and DU show constant power consumption while UE has an increase in power consumption when the traffic is introduced. The averages of power and energy consumption for all devices for all baselines can be seen in table 4.1.

Table 4.1: Average power and energy consumption of the devices for three different baselines

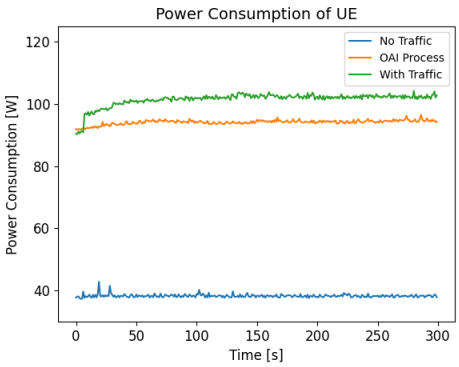
Case	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
No Traffic	52.21	57.64	38.31	15610.21	17234.49	11454.54
OAI Process	106.45	116.36	94.10	31828.18	34791.99	28137.36
Traffic	108.14	119.47	101.41	32333.40	35723.39	30326.59



(a) Power consumption of CU



(b) Power consumption of DU



(c) Power consumption of UE

Figure 4.3: Power consumption of the devices - After separating UE from the CU

4.1.3 Obtaining baselines when the processor is 100% active

In one of the stages of the work some changes to the testbed were made. One of those changes was disabling hyper-threading. This was suggested by the OpenAirInterface community [20]. Although this was an optional setting, we chose to evaluate it as well. We concluded that disabling hyper-threading did not impact the network in any way, the power consumption stayed the same. On the other hand, second change that was made was setting the machines to run only in C0 state. Modern processors feature various power-saving modes known as C-states. These modes enable an idle processor to shut down unused components, thereby conserving power. The higher the number of C-states the more components will get shut down and more power would get saved. The overview of the states can be seen in table 4.2.

By fixing the processor to C0 operational state, it is ensured that the CPU is fully turned on and the core is in one of the performance states. This is a good solution for increasing performance, but it also causes an increase in power consumption, which was the situation in our case. Therefore, new measurements were conducted to understand the updated power baselines. We are again looking at the total power consumption in three cases. The results can be seen in figure 4.4. Table 4.3 shows the average power and energy consumption for the devices correspondent to these

Table 4.2: Processor C-states, from [27]

Mode	Definition
C0	Operational state in which the CPU is fully turned on.
C1	The first idle state stops the CPU's main internal clocks via software control, but the bus interface unit and APIC remain fully operational.
C2	In this state, the CPU's main internal clocks are stopped via hardware. Although the processor maintains all software-visible states, it may take longer to wake up in response to interrupts.
C3	In this state, all CPU internal clocks are stopped. The processor does not need to maintain cache coherence but retains other states. Some processors have variations of the C3 state that differ in the time required to wake the processor in response to interrupts.

changes.

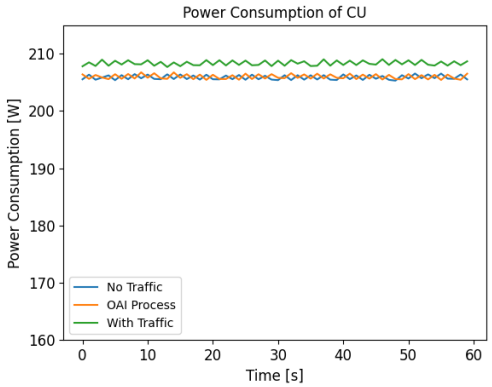
Table 4.3: Average total power consumption and energy consumption of the devices for three different cases when processor is fully activated

Case	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
No Traffic	205.87	196.39	175.25	12146.60	11586.68	10339.41
OAI Process	205.99	201.62	172.02	12152.88	11895.86	10149.06
Traffic	208.35	203.71	170.57	10149.06	12019.35	10064.36

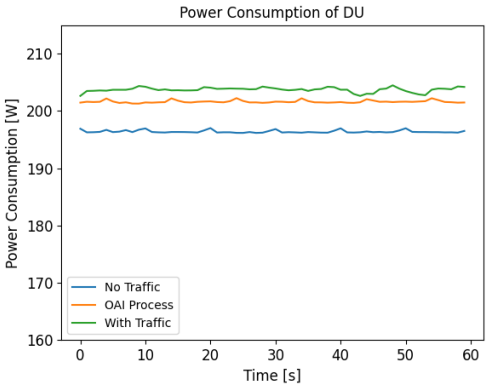
4.2 Dependency of Power Consumption on Data Rate

As previously stated, one of the main objectives of this work was to assess network performance in terms of both throughput and power consumption. For this analysis, as outlined in 4.1, we considered three scenarios to compare differences between a static case with no traffic and a dynamic case with traffic. For the dynamic case, it was decided to input UDP traffic at the rates of 8Mbps, 16Mbps, 32Mbps and 64Mbps. All of the measurements were performed for the time of 5 minutes. For these measurements the hyper-threading was enabled and the machines were running using all processor core states.

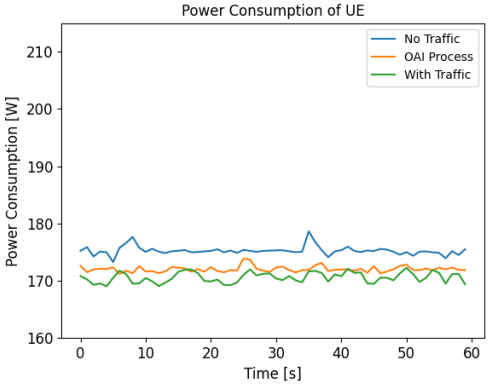
The results show the throughput at the client (Core) and server (UE) side as well as the power consumption of the DU and UE. Since we only have one UE device, one client, we did not expect any significant strain in the CU. For this reason, we have chosen to plot only DU and UE power consumption, UE being the receiver of the traffic, and DU for it's significant role of handling baseband processing, such as modulation/demodulation, encoding/decoding and more. However, in the table 4.4 we presented all three devices and their average power and energy consumption. As for the plots, the metric used for obtaining the results focuses only on the excess power consumption over the OAI process baseline power consumption, measured in section 4.1.2 and shown in table 4.1. This way, only the power consumption caused solely by the traffic can be observed.



(a) Power consumption of CU



(b) Power consumption of DU



(c) Power consumption of UE

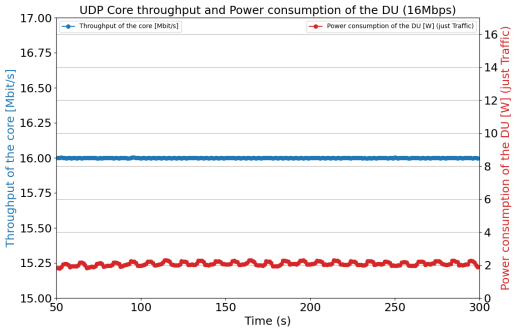
Figure 4.4: Total power consumption of the devices when processor is fully activated

The results for each data rate are shown in the next figures 4.5 - 4.8. Also, the table 4.5 shows the average obtained throughput from these measurements.

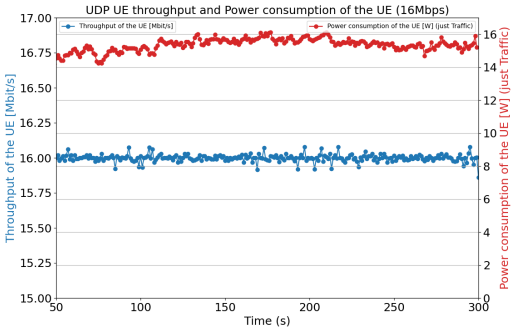
Upon examining the results, it is evident that changes in data rate do not significantly affect power consumption; however, there are still minor variations in power consumption. When examining the UE specifically, it is important to note that the variation in power consumption from 32 Mbps to 64 Mbps is not as significant as the variation from 8 Mbps to 16 Mbps and from 16 Mbps to 32 Mbps, which was unexpected. However, there are notable variations in throughput for data rates of 32Mbps and 64Mbps. Specifically, at 64Mbps, the achieved throughput is much lower than expected. Due to this, further investigation was required.

Table 4.4: Average power and energy consumption of the devices for different rates of UDP traffic

Data rate [Mbps]	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
8	107.48	119.16	101.77	32242.62	35748.77	30531.35
16	107.80	120.04	108.14	32341.75	36012.91	32441.61
32	107.92	120.89	113.58	32376.97	36267.42	34074.40
64	108.45	121.40	115.14	32535.70	36419.36	34543.34

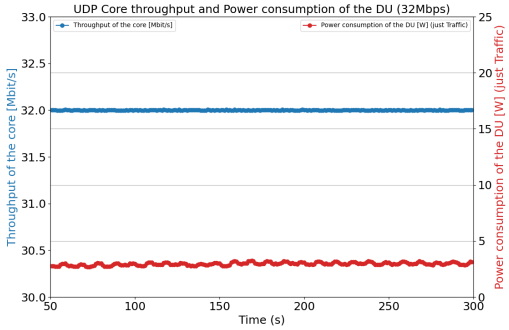


(a) Throughput of the core vs excess over average OAI process power consumption of DU

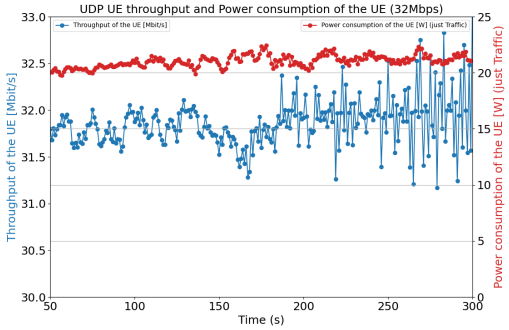


(b) Throughput of the UE vs excess over average OAI process power consumption of UE

Figure 4.6: Throughput vs excess over average OAI process power consumption for UDP traffic of 16Mb

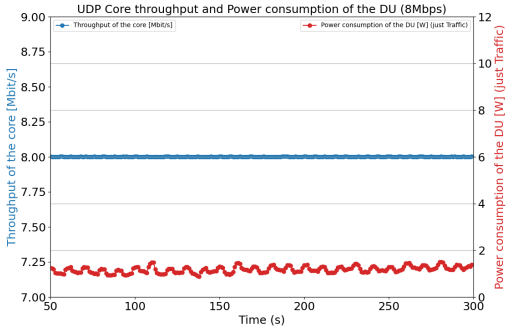


(a) Throughput of the core vs excess over average OAI process power consumption of DU

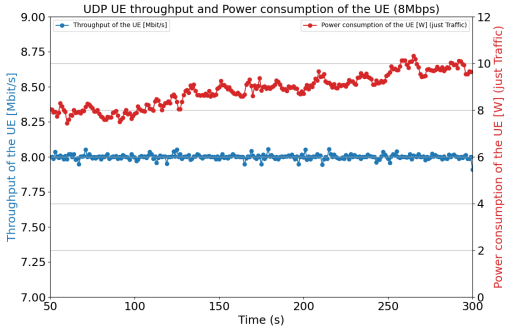


(b) Throughput of the UE vs excess over average OAI process power consumption of UE

Figure 4.7: Throughput vs excess over average OAI process power consumption for UDP traffic of 32Mb

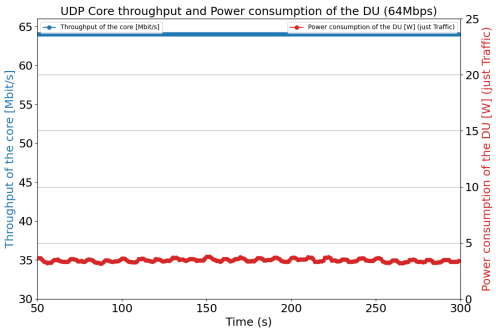


(a) Throughput of the core vs excess over average OAI process power consumption of DU

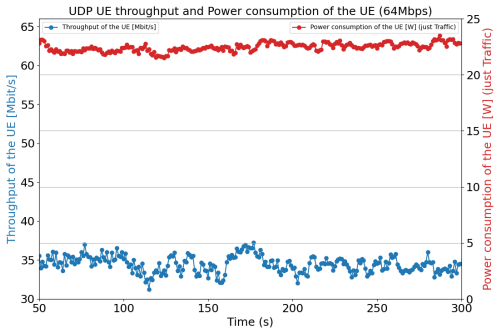


(b) Throughput of the UE vs excess over average OAI process power consumption of UE

Figure 4.5: Throughput vs excess over average OAI process consumption for UDP traffic of 8Mb



(a) Throughput of the core vs excess over average OAI process power consumption of DU



(b) Throughput of the UE vs excess over average OAI process power consumption of UE

Figure 4.8: Throughput vs excess over average OAI process power consumption for UDP traffic of 64Mb

Table 4.5: Average achieved throughput and standard deviation at the server side for different rates of UDP traffic on the client side

Side	Average throughput [Mbps] | Standard deviation [Mbps]			
Client	8 | 0	16 | 0.01	32 | 0.01	64 | 0.01
Server	8 | 0.08	16 | 0.01	31.87 | 0.11	34.47 | 2.36

4.2.1 Understanding Throughput Limitations for UDP traffic

During subsequent measurements with a 64Mbps data rate, an error message was observed:

```
/home/du/openairinterface5g-marija-jocic/openair2/LAYER2/nr_rlc/
nr_rlc_entity_am.c:1769:nr_rlc_entity_am_recv_sdu:
warning: 237 SDU rejected, SDU buffer full
```

By examining the error message, it is evident that the error occurred in the Service Data Unit (SDU) buffer. RLC (Radio Link Control) protocol layer in which this error occurred is, as explained in section 2.1.3 the layer that provides transmission of data segments. It provides three modes of operation [26]:

- Acknowledged mode (AM) - receiver informs the sender if the packets are received correctly or not and triggers retransmission if needed.
- Unacknowledged mode (UM) - stores packets in send buffer; sender is not aware if the packets were successfully received. It is possible to segment the packets before transmission and reassembly them on the receiving side.
- Transparent mode (TM) - only stores packets in a send buffer; sender is not informed of the state of the received packet.

Furthermore, when the RLC receives a data unit from the upper layer, PDCP, it encapsulates it as an RLC SDU and stores it in the **RLC SDU buffer** [18]. It divides the SDU into RLC Protocol Data Units (PDUs), ensuring the use of each PDU. SDU can be placed within one PDU or spread across several PDUs, and a PDU can contain segments from multiple SDUs. These PDUs are stored in a dedicated buffer, waiting for an opportunity to be transmitted to a MAC layer. In RLC AM, a PDU is removed from a buffer only after receiving an acknowledgment. An RLC SDU is removed from the buffer only after all its segments have been transmitted from all PDUs.

Knowing all this led us to confirm the configuration of our RLC. It turned out that we were dealing with RLC AM. First solution we had, was to increase the SDU buffer size. To do this RLC_RX and RLX_TX buffers were increased, in the source code configuration, and measurements were repeated for the data rate of 64Mbps. At first glance, it appeared that the problem was resolved, as we managed to sustain the traffic at this rate for a 5 minute measurement period. However, when the traffic was reintroduced for another 5 minutes, the buffer filled up again. It was concluded that this solution only partially solved the issue, as the buffer still became full, but at a different time.

Next step was to consult the OpenAirInterface developers via their mailing list. They confirmed that this problem arises due to too much traffic being sent, causing the internal RLC buffers to fill up and data to be dropped. They also confirmed that this issue is still in the development. The proposed solution was to monitor the availability of the SDU buffer, more concretely to monitor the buffer getting empty. This was to be done by looking at the Logical Channel ID (LCID) in the real-time logs. Concretely, one of the LCIDs was reporting on the user plane data. The moment this data would stop increasing would mean that the buffer got empty. To monitor this, we began to record real-time logs while the traffic of 64Mb was flowing. However, this led us to a new path. By examining these logs, we noticed a high number of retransmissions in the MAC layer. There were four rounds of transmissions. First was correspondent to the number of transmissions. Second, third and fourth were pointing out to the retransmissions. The recorded number of retransmissions for the throughput of 64Mbps can be observed in figure 4.9. Based on these results, we can notice

how when the number of retransmissions increases the throughput at the UE decreases. For this reason, we suspect that the processing on the receiving side, UE, is not fast enough due to this high amount of retransmissions. To fully address this issue, using commercial UEs would be the solution, as their response times are typically shorter, leading to higher throughput.

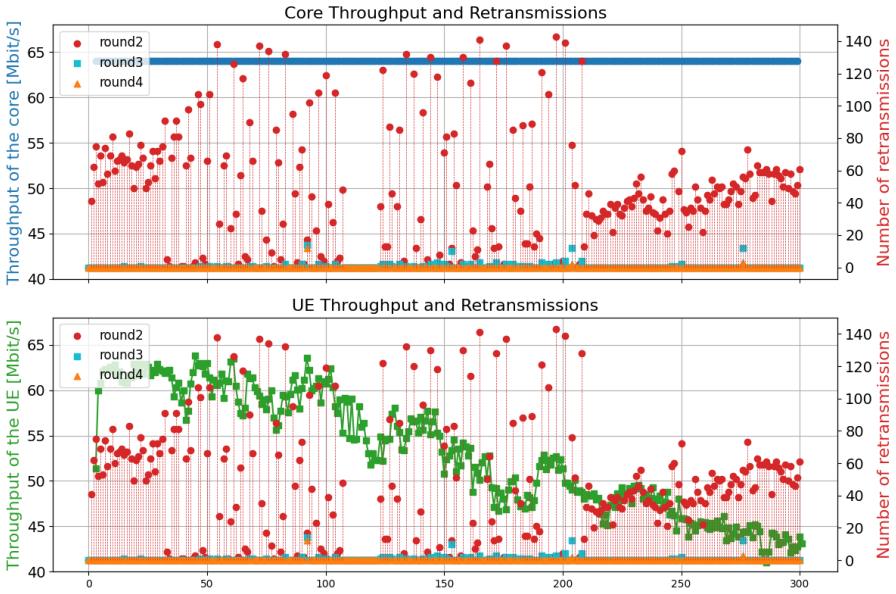
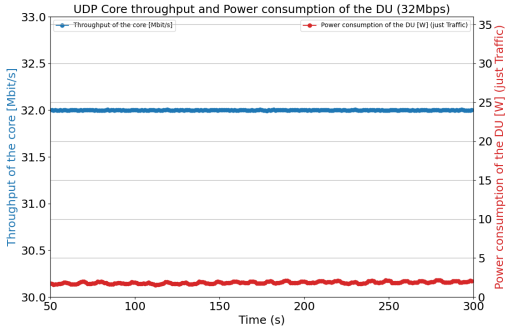
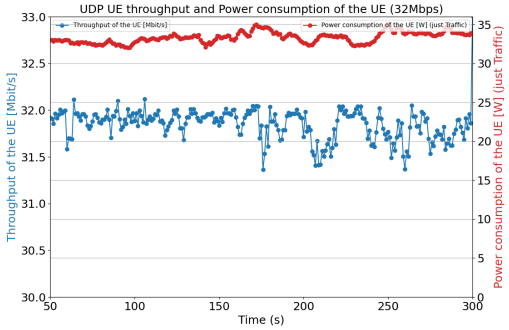


Figure 4.9: Throughput of the core and UE for UDP traffic, and the number of retransmissions in 3 rounds

Nevertheless, to get to the final measurements, when monitoring the buffer it was possible to obtain the data rate of 64Mbps. This can be consulted in figures 4.10 and 4.11 for traffic of 32Mbps and 64Mbps, respectively. Still, these results do not offer a clear picture of the achievable throughput because, in a real-world scenario, traffic would not be limited to just 5 minutes. Additionally, the throughput requirement would be much higher than 64 Mbps. But, for our testbed, it would be more advisable to adhere to values lower than 64 Mbps.

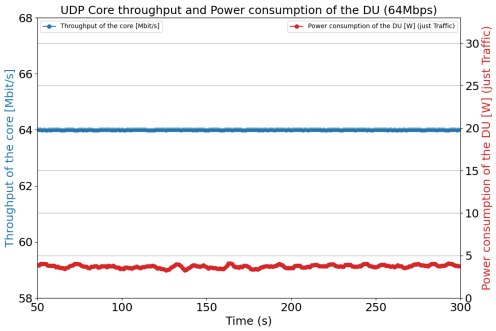


(a) Throughput of the core vs excess over average OAI process power consumption of DU

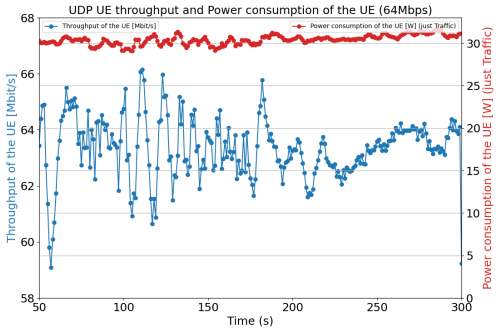


(b) Throughput of the UE vs excess over average OAI process power consumption of UE

Figure 4.10: Throughput vs excess over average OAI process power consumption for UDP traffic of 32Mb while monitoring the buffer



(a) Throughput of the core vs excess over average OAI process power consumption of DU



(b) Throughput of the UE vs excess over average OAI process power consumption of UE

Figure 4.11: Throughput vs excess over average OAI process power consumption for UDP traffic of 64Mb while monitoring the buffer

4.2.2 Understanding Throughput Limitations for TCP traffic

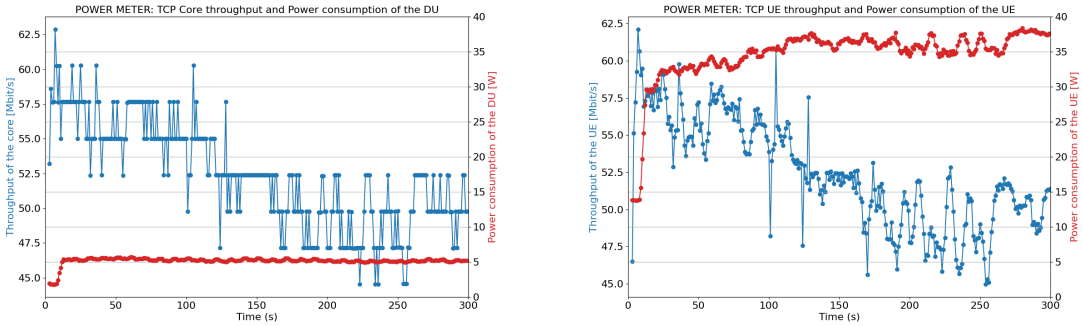
To better understand and support the findings regarding the throughput limitations explained in section 4.2.1, we conducted a similar research only for the TCP traffic.

In this case, the TCP was not limited to any number of bits, and as before the measurements were conducted for the time of 5 minutes. Here we were also focused on the excess power over the average OAI baseline power to present the results. The results can be seen in figure 4.12. It can be confirmed once again that throughput does not significantly impact power consumption. For the DU, power consumption remains constant, while for the UE, there are more variations, but they are not extreme. The average power and energy consumption of the total power and energy consumption can be seen in table 4.6. As for the throughput, the throughput of the core tends to adapt to the throughput of the UE, which is a characteristic of TCP traffic. The throughput

stabilises around 48-50Mbps. Based on this, we can assume that this is the data rate at which the SDU buffer would not get full and we can operate freely. In figure 4.13 we can also see the number of retransmissions for TCP traffic. It is clear that the number of retransmissions increase in the moments of considerable changes in the throughput.

Table 4.6: Average power and energy consumption of the devices for TCP traffic

Traffic type	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
TCP	111.18	121.52	128.68	33353.63	36456.71	38603.68



(a) Throughput of the core vs excess over average OAI process power consumption of DU

(b) Throughput of the UE vs excess over average OAI process power consumption of UE

Figure 4.12: Throughput vs excess over average OAI process power consumption for TCP traffic

In addition, we were interested to see how will the throughput behave in the case of core being in C0 state, when the performance should be at its maximum. The configuration was changed and new measurements performed. The results can be seen in figure 4.14 regarding the power consumption and throughput and figure 4.15 regarding the retransmissions for the obtained throughput. Table 4.7 also shows the average power and energy consumption in this case. The power consumption is increased as expected, where the DU has constant power consumption, but UE now experiences higher variations. Since these measurements take into account the whole device, we cannot be sure what would cause such variation. We can only assume that there was another system component, e.g. a fan, that stopped consuming so much power at this time before turning on again. Anyway, what is intriguing is that in this scenario, where improved performance is expected, the throughput is lower, stabilizing around 40 Mbps.

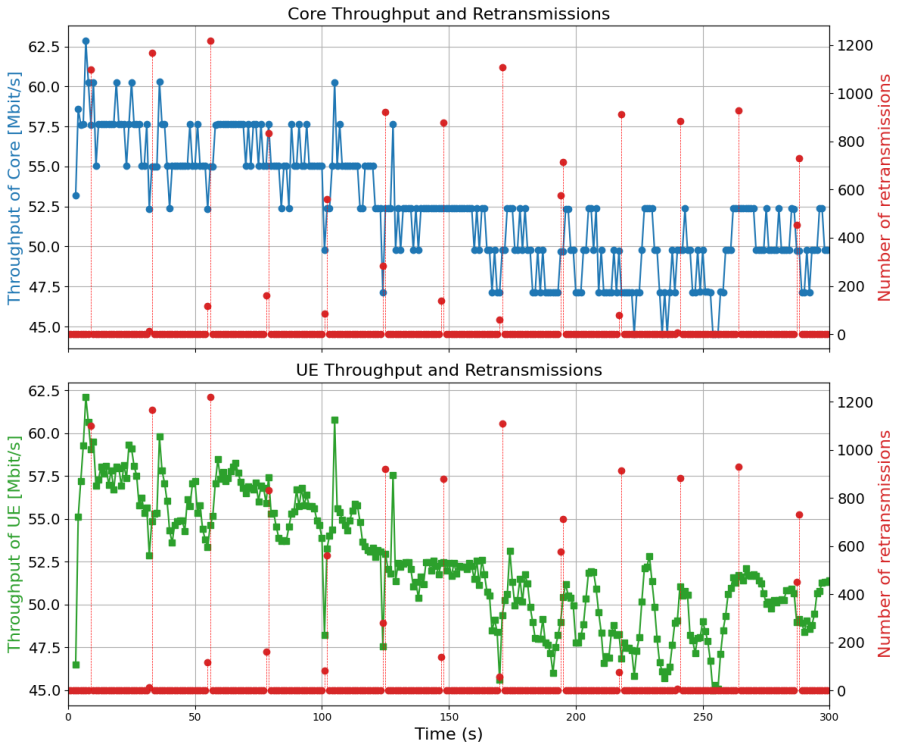
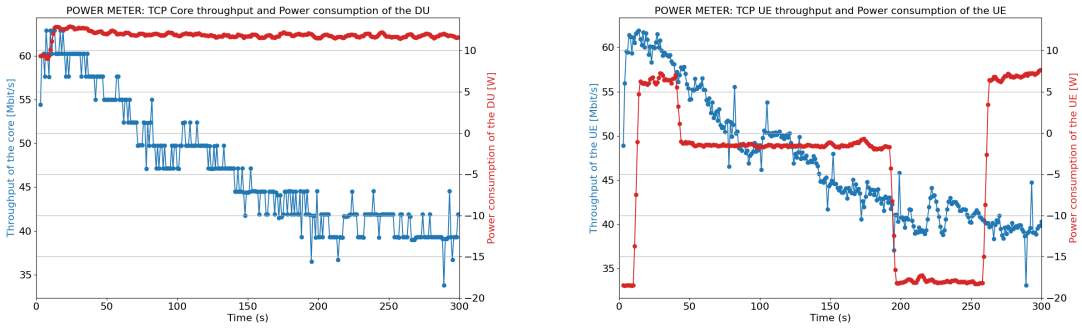


Figure 4.13: Throughput of the core and UE for TCP traffic and the number of retransmissions



(a) Throughput of the core vs excess over average OAI process power consumption of DU when the core is in C0 state

(b) Throughput of the UE vs excess over average OAI process power consumption of UE when the core is in C0 state

Figure 4.14: Throughput vs excess over average OAI process power consumption for TCP traffic when the core is at C0 state

Table 4.7: Average power and energy consumption of the devices for TCP traffic when the core is in C0 state

Traffic type	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
TCP	211.21	213.49	168.20	63363.94	64047.75	50459.63

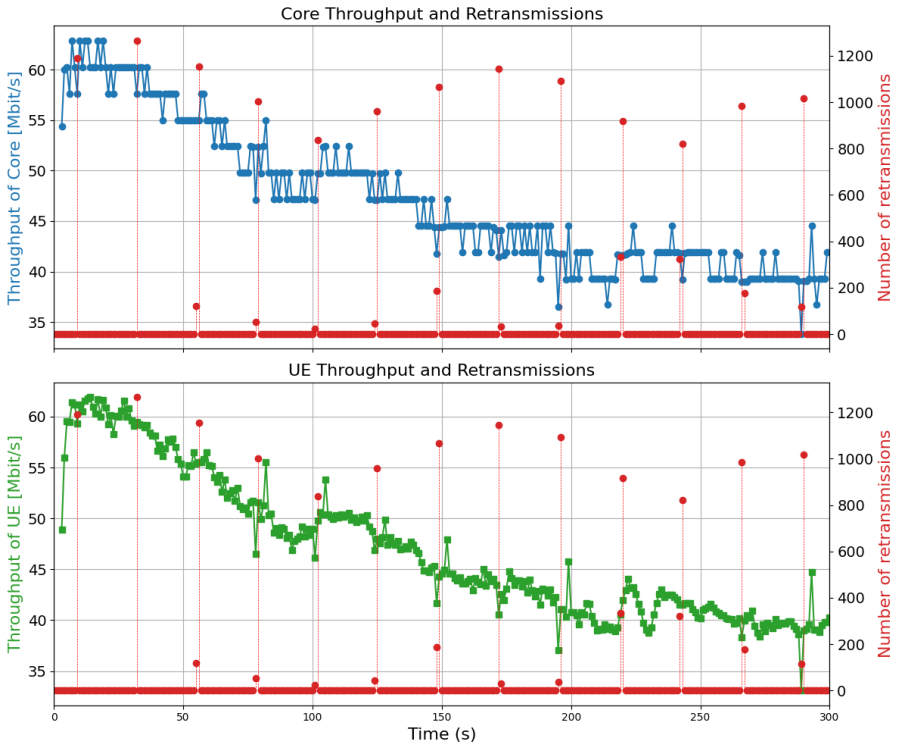


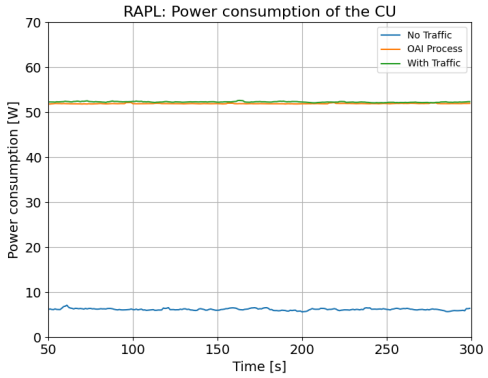
Figure 4.15: Throughput of the core and UE for TCP traffic and the number of retransmissions when the core is in C0 state

4.3 Assessing Processor Share in Power Consumption

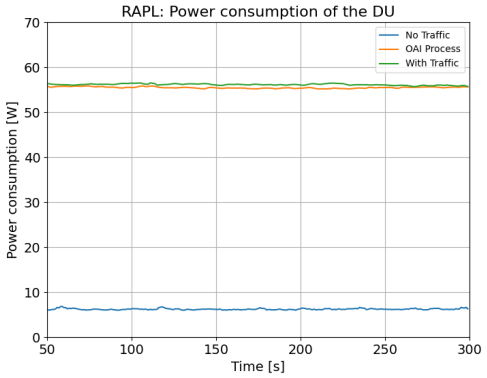
4.3.1 Fraction of processor power in total power

In this section we are trying to understand the portion of processor power in the total power consumption. To measure the power consumption of the processor, we used RAPL tool. As explained in section 3.4, RAPL has several domains. The one we were interested in was Package domain.

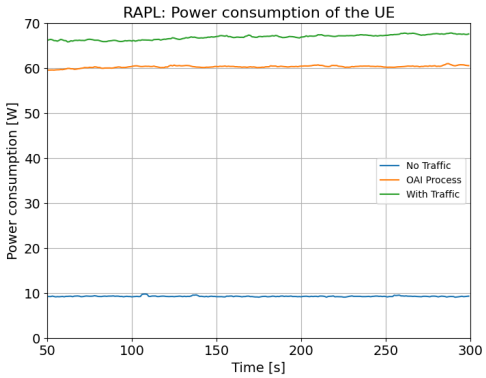
By simultaneously executing *turbostat* command explained in section 3.4 on CU, DU and UE machines, the results were obtained. They are shown in figure 4.16. Also, the average power consumption and energy consumption of each device can be consulted in table 4.8. What can be seen is that the UE consumes more power than the CU and DU. This is because the UE, as the receiver of traffic, performs highly computational decoding functions. When comparing these results to those shown in figure 4.1, it becomes apparent that a significant portion of the power consumption is attributed to processor usage in cases of OAI process running and having traffic, in both CU and DU scenarios, with UE exhibiting even higher levels of processor-related power consumption. This can again be attributed to the UE's functionality. Results can be consulted in table 4.9.



(a) Power consumption of CU's processor



(b) Power consumption of DU's processor



(c) Power consumption of the UE's processor

Figure 4.16: CPU power consumption of the devices

Table 4.8: Average processor power and energy consumption of the devices for three different cases

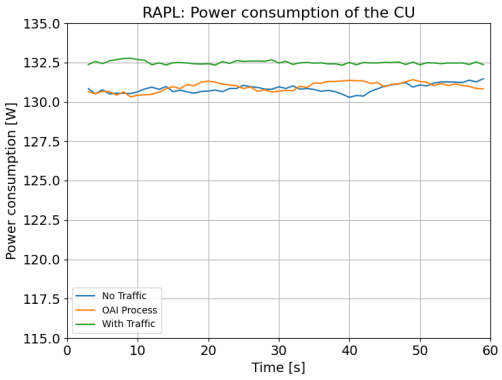
Case	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
No Traffic	6.16	6.21	9.27	1849.32	1862.99	2789.64
OAI Process	51.87	55.44	60.14	15544.70	16615.82	17990.77
Traffic	52.24	56.08	66.61	15656.04	16810.34	19984.88

Table 4.9: Percentage of processor power consumption in total power consumption

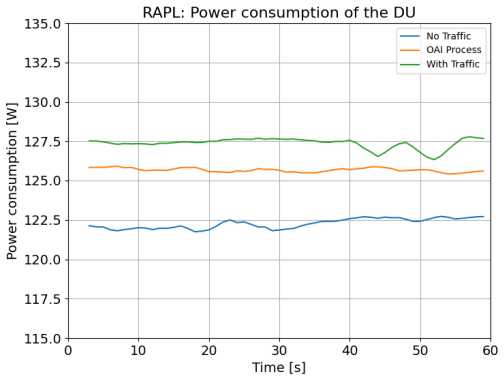
Case	(Processor / Total from table 4.1) power consumption [%]		
	CU	DU	UE
No Traffic	11.80	10.77	24.20
OAI Process	48.73	47.64	63.91
Traffic	48.31	46.94	65.68

4.3.2 Fraction of processor power in total power when processor is 100% active

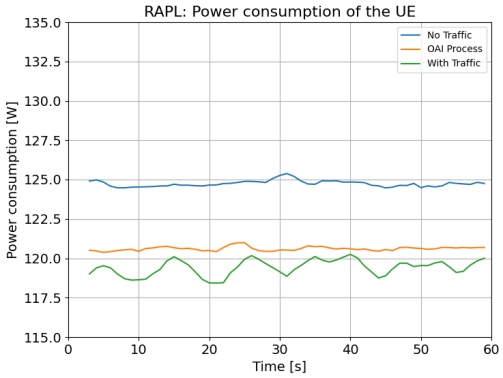
As mentioned in section 4.1.3 in one of the stages of the work some changes to the testbed were made. These changes involved disabling hyper-threading and setting machines to run only in C0 state. This made an impact on the processor power and energy consumption which can be seen in figure 4.17. Also, table 4.10 shows the new values for average power and energy consumption per device and table 4.11 shows the percentage of processor power attributable to the total power consumption. As expected, with the processor now fully activated, its power consumption has increased significantly, becoming a larger fraction of the total power consumption.



(a) Power consumption of CU's processor



(b) Power consumption of DU's processor



(c) Power consumption of the UE's processor

Figure 4.17: CPU power consumption of the devices when processor is fully activated

Table 4.10: Average processor power and energy consumption of the devices for three different cases when processor is fully activated

Case	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
No Traffic	130.86	122.27	124.74	7727.67	7824.67	7367.44
OAI Process	130.95	125.67	120.61	7733.59	7422.02	7123.07
Traffic	132.48	127.37	119.40	7824.67	7522.24	7051.93

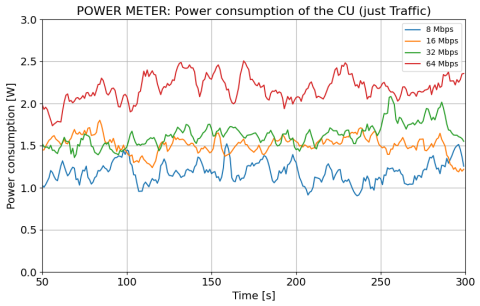
Table 4.11: Percentage of processor power consumption in total power consumption when processor is fully activated

Case	(Processor / Total from table 4.3) power consumption [%]		
	CU	DU	UE
No Traffic	63.56	62.26	71.18
OAI Process	63.57	62.33	70.11
Traffic	63.59	62.53	70.00

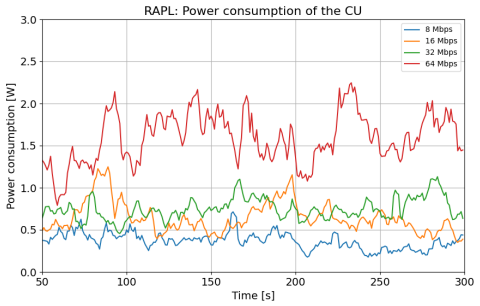
4.3.3 Dependency of CPU's power consumption on Data Rate

In section 4.2, the goal was to understand how is the power consumption dependent of the changes in traffic. In this section similar analysis is conducted, but with a focus on processor power and fraction of processor power on total consumption. The aim is to determine how traffic changes influence processor power, and to further analyze the correlation between total power consumption and processor power consumption.

For consistency, measurements were performed simultaneously on the power meter and with RAPL. They took length of 5 minutes and UDP traffic in rates of 8, 16, 32 and 64Mb was introduced. For this hyper-threading was enabled and core was running in all states. As before, for presenting the results, we were only considering the excess power over the average of OAI Process baseline power. The plots showing the power that was measured can be consulted in figures 4.18 to 4.20. Furthermore, in table 4.12 average energy consumption for different data rates can be observed. Here we are presenting the total energy consumption for the CU, DU and UE processor. Comparing this to the table 4.4 and by examining the figures 4.18 to 4.20, we can conclude that there are other components, such as memory, fan etc., whose power consumption depends on the data rate, more than the CPU. This is because the total power changes with data rate, but the CPU power not very significantly, i.e. the CPU power only explains the total power in part. The precise portion of processor power in total power can be seen in table 4.13.

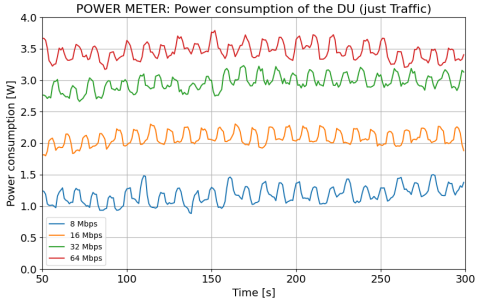


(a) Excess over average OAI process baseline power consumption of CU for different UDP data rates measured on power meter

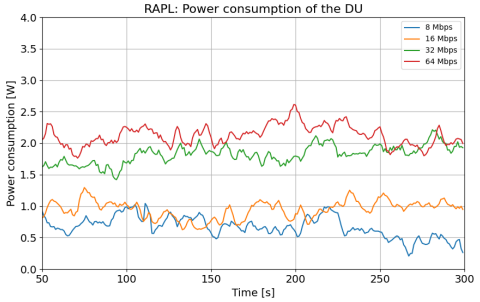


(b) Excess over average OAI process baseline power consumption of CU's CPU for different UDP data rates measured with RAPL

Figure 4.18: Excess over average OAI process baseline power consumption of the CU - Total vs Processor



(a) Excess over average OAI process baseline power consumption of DU for different UDP data rates measured on power meter

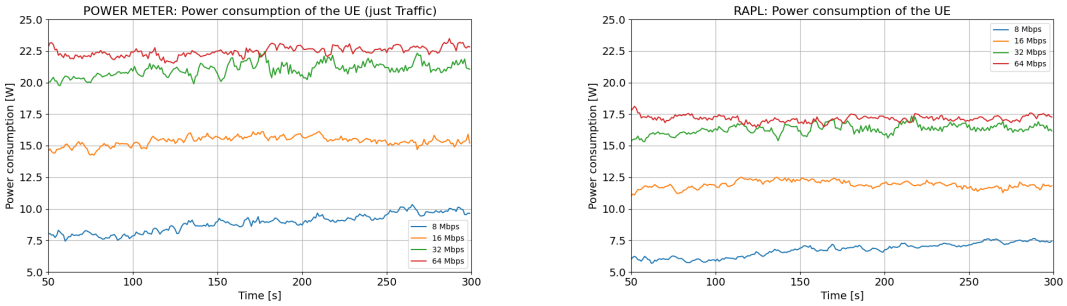


(b) Excess over average OAI process baseline power consumption of DU's CPU for different UDP data rates measured with RAPL

Figure 4.19: Excess over average OAI process baseline power consumption of the DU - Total vs Processor

Table 4.12: Average processor energy consumption of the devices for different data rates of UDP traffic

Data rate [Mbps]	Average power consumption [W]			Average energy consumption [J]		
	CU	DU	UE	CU	DU	UE
8	52.19	56.03	66.61	15656.04	16810.34	19984.88
16	52.42	56.29	71.66	15725.24	16888.30	21496.92
32	52.52	57.09	75.93	15757.22	17127.10	22778.15
64	53.30	57.45	76.95	15990.50	17235.08	23084.12



(a) Excess over average OAI process baseline power consumption of UE for different UDP data rates measured on power meter

(b) Excess over average OAI process baseline power consumption of UE's CPU for different UDP data rates measured with RAPL

Figure 4.20: Excess over average OAI process baseline power consumption of the UE - Total vs Processor

Table 4.13: Percentage of power consumption in total power consumption for different data rates of UDP traffic

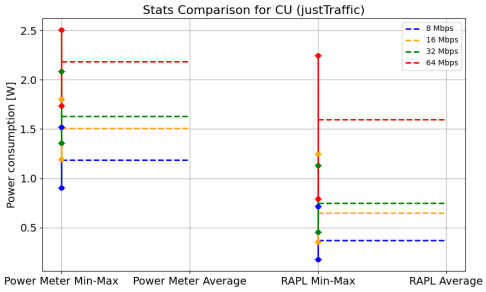
Data rate [Mbps	(Processor / Total) power consumption [%]		
	CU	DU	UE
8	48.58	47.02	65.45
16	48.63	46.89	66.14
32	48.67	47.22	66.85
64	49.15	47.32	66.83

To increase the clarity of throughput effects and facilitate comparison between total power consumption and processor power consumption, the results can be presented in statistical terms. We chose to present the minimum, maximum and average value of power consumption for each data rate and device. This is shown in figure 4.21.

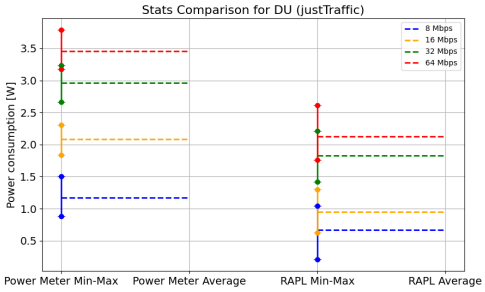
What can be seen is that the effect of data rate on CU is rather small. If we look at the average values from the case of having 8Mbps to case of 64Mbps, total power consumption changed by 1 Watt, while the processor power consumption changed by a little more then 1 Watt. Now, looking at the DU, the variation is just a bit bigger. For total power it is about 2 Watts, while for processor power even less than 2 Watts. Lastly, the variation on the UE is higher than in former two cases. Total power changed by approximately 13 Watts and the processor power by 11 Watts. Based on this it can be concluded that the data rate does not have a significant effect on the power consumption in general, except in case of UE, which can be attributed to the fact that the UE is the receiver of traffic and has to perform decoding of data.

After this analysis, besides confirming once again the minor impact of traffic on power consumption, we observed that the results obtained at 64 Mbps are quite similar to those at 32 Mbps. This reaffirms the throughput limitations, indicating that the actual achievable throughput is around

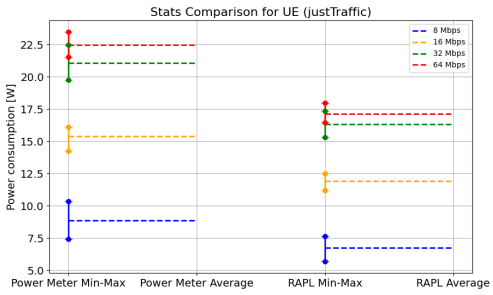
40 Mbps, explaining why the results align more closely with the 32 Mbps scenario. This is especially noticeable for the UE.



(a) CU statistics for excess power over the average for total and processor power consumption



(b) DU statistics for excess power over the average for total and processor power consumption



(c) UE statistics for excess power over the average for total and processor power consumption

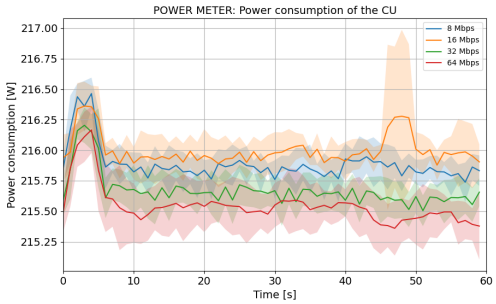
Figure 4.21: Statistics of excess power consumption over average for the total power and processor power of the devices

4.4 Data rate effect on CPU performance counters

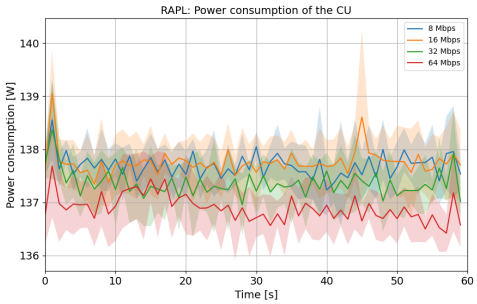
Exploring the following topic of CPU performance counters is necessary because total power consumption is not solely dependent on the processor and data rate, which was deduced in the previous sections. Therefore, we are examining processor events and memory in greater detail to better understand the power consumption.

For this analysis hyper-threading was disabled and the core was fixed to only C0 state in order for the CPU to be fully activated. Fully activating the CPU by disabling hyper-threading and fixing the core state to C0 allows for accurate measurement and assessment of the processor's true capabilities under maximum workload conditions, ensuring reliable performance evaluations and identifying any potential bottlenecks or limitations. This was crucial because it allowed us to thoroughly explore all functionalities of the processor. Furthermore, the experiments described in section 4.3.3 with different data rates of UDP were repeated, yielding the same conclusions, except the values being in a higher range. Results of new measurements are in figures 4.22, 4.23

and 4.24. Here we were looking at the total power consumption. The shaded area indicates the deviation. CU and DU display constant power consumption and UE expresses a certain variations.

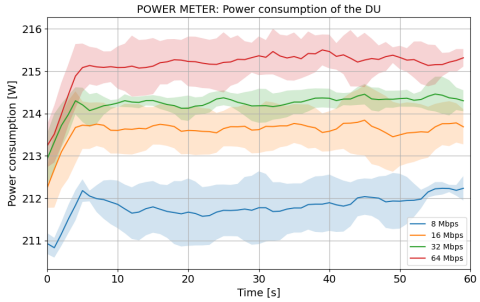


(a) Power consumption of CU for different data rates measured on power meter

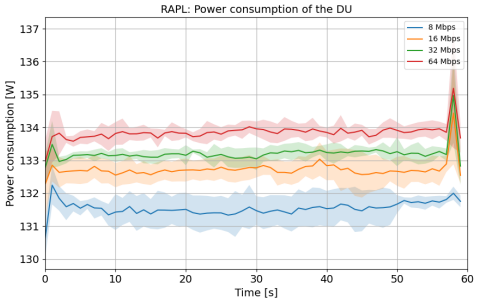


(b) Power consumption of CU's CPU for different data rates measured with RAPL

Figure 4.22: Power consumption of the CU - Total vs Processor

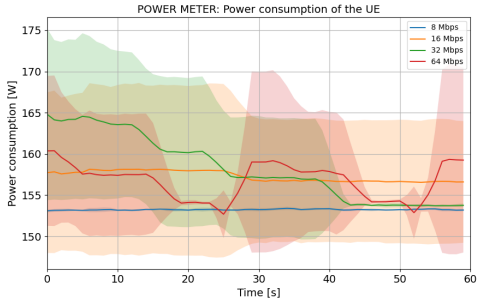


(a) Power consumption of DU for different data rates measured on power meter

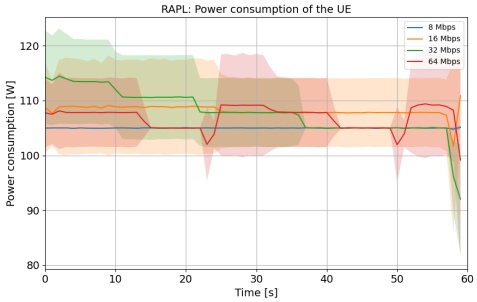


(b) Power consumption of DU's CPU for different data rates measured with RAPL

Figure 4.23: Power consumption of the DU - Total vs Processor



(a) Power consumption of UE for different data rates measured on power meter



(b) Power consumption of UE's CPU for different data rates measured with RAPL

Figure 4.24: Power consumption of the UE - Total vs Processor

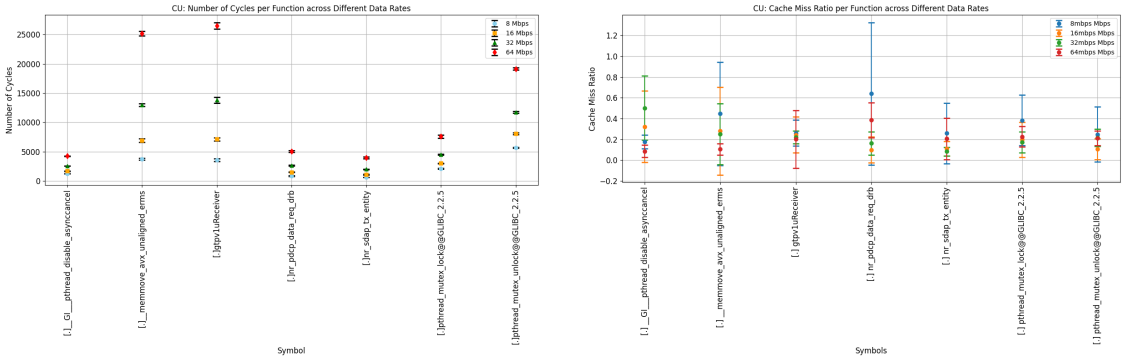
4.4.1 Number of cycles and Cache miss ratio changes due to the data rate

At this stage of our research, we began delving deeper into understanding the factors influencing power consumption and overall performance. Our approach involved monitoring processor events to identify which functions within the processes of our network have the most significant impact on power usage. Additionally, we investigated whether data rates influence these events. Specifically, we focused on cache misses and cache references, crucial for calculating the cache miss ratio across different functions.

Furthermore, we documented the number of cycles each function required to prioritize those of greater importance. Examining how data rates affect cycles, cache misses, and cache references is essential for understanding and improving code performance. By recognizing patterns, developers can potentially rewrite or restructure code to minimize cache misses and enhance overall efficiency. In addition, analyzing these metrics provides valuable insights into how effectively hardware resources are utilized and identifies opportunities for optimizing resource allocation.

The *perf* tool explained in section 3.4, generated a readable output file listing all functions related to the desired events. Upon reviewing this file, we observed that certain functions exhibited significant overhead in terms of cycles per period. These functions were of particular interest to us, as we concluded those must be the functions causing the highest power consumption. Therefore, we focused our analysis on functions where the overhead exceeded 1% and which appeared for all data rates and in the cache misses and cache references lists.

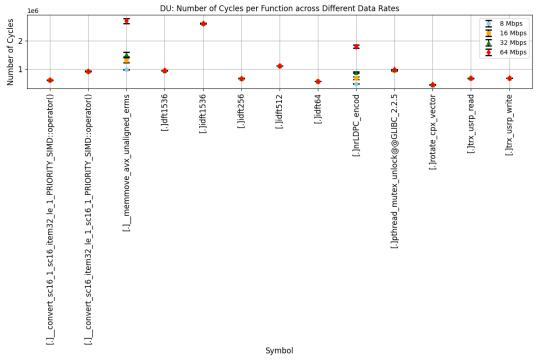
For these functions, we proceeded to calculate cache miss ratios to further understand their performance characteristics. This approach allowed us to pinpoint specific areas where optimizing cache usage could potentially lead to efficiency improvements and reduced power consumption. The resulting plots for each device and different data rates can be observed in figures 4.25, 4.26 and 4.27.



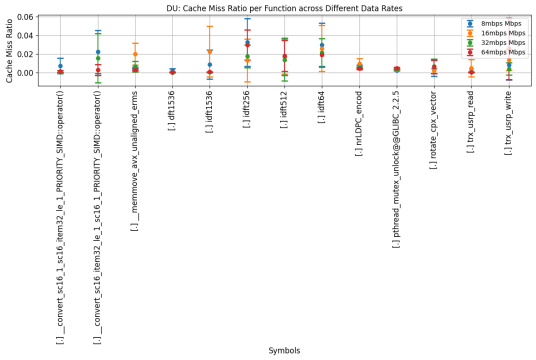
(a) Number of cycles per function of the CU for different data rates

(b) Cache miss ratio per function of the CU for different data rates

Figure 4.25: Number of cycles and Cache miss ratio per function of the CU for different data rates

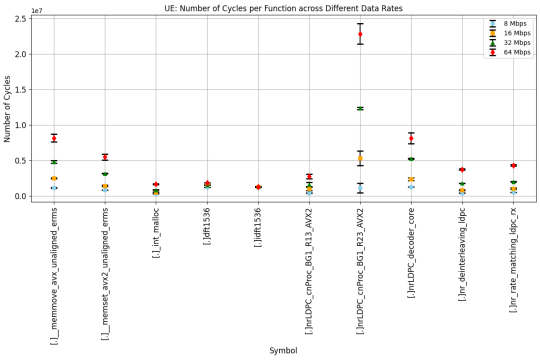


(a) Number of cycles per function of the DU for different data rates

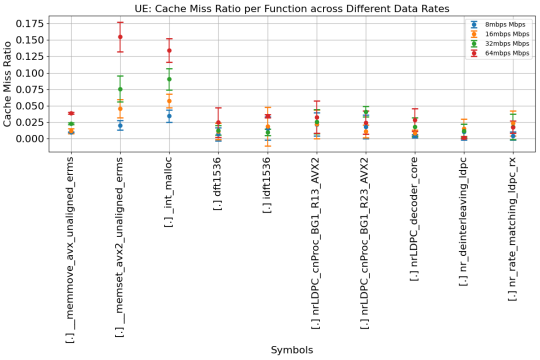


(b) Cache miss ratio per function of the DU for different data rates

Figure 4.26: Number of cycles and Cache miss ratio per function of the DU for different data rates



(a) Number of cycles per function of the UE for different data rates



(b) Cache miss ratio per function of the UE for different data rates

Figure 4.27: Number of cycles and Cache miss ratio per function of the UE for different data rates

After obtaining the results we decided to focus on even smaller number of functions since only a few showed increase in number of cycles when changing the data rate. Taking this into account, the extracted functions for the CU were: *__memmove_avx_unaligned_erms*, *gtpv1uReceiver* and *pthread_mutex _unlock@@GLIBC_2.2.5*. For the DU, we focused on *__memmove_avx_unaligned _erms* and *nrLDPC_encod*, and for the UE we had *__memmove_avx_unaligned_erms*, *nrLDPC _cnProc_BG1_R23_AVX2* and *nrLDPC_decoder_core*.

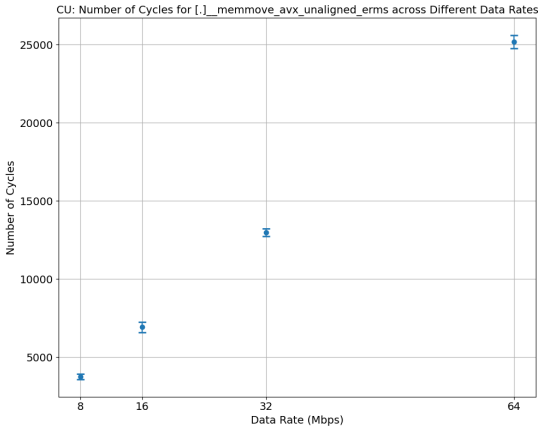
Interpreting the results, noticeable functions are the ones related to New Radio LDPC (Low Density Parity Check) encoding and decoding on the traffic channel. Moreover, GTP-U receiver, on the CU machine, due to the exchange of data between the RAN and core network reports changes in number of cycles. Since both of these functions seem related to the traffic, we would assume that these functions are the ones affected by the changes in the data rate.

Interestingly enough, *memmove*, which is a memory move function, was selected for all three devices. Also, *pthread_mutex _unlock* was selected on the CU device. To understand if these

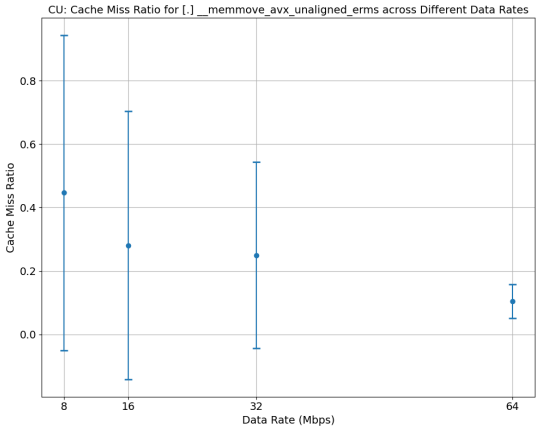
functions are affected by the data rate, we would need to know which higher function calls these functions. But, based on results, they do seem to be influenced by traffic taking into account the increase in number of cycles.

As for the cache miss ratio, all of the functions seem to be having an overall low cache miss ratio, which is a sign of solid performance of the network. As a result, our findings suggest that while certain functions are indeed influenced by data rate, the overall performance and power consumption do not appear to be significantly affected by it. However, these functions could have a substantial effect at higher data rates due to their increasing number of cycles. As future steps would involve working with higher data rates, it is essential to be aware of these functions and monitor their behavior closely. Ensuring optimal performance at higher data rates will be essential for advancing our network capabilities. Plots showing each of the function's statistics are shown in figures 4.28 - 4.35.

CU functions:

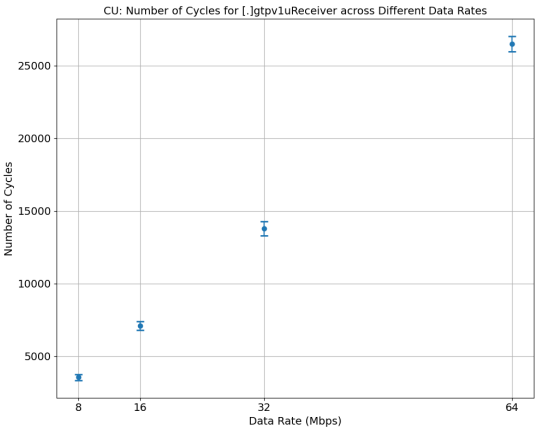


(a) Number of cycles for memory move function of the CU for different data rates

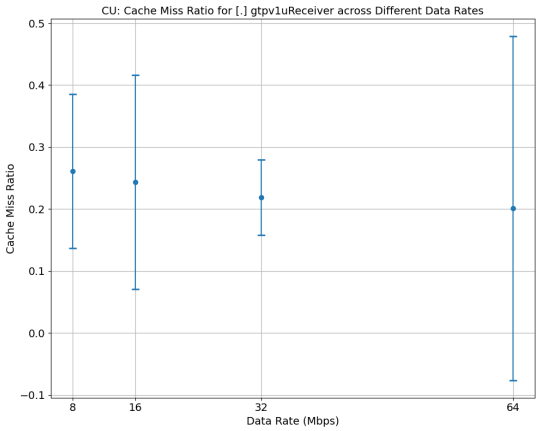


(b) Cache miss ratio for memory move function of the CU for different data rates

Figure 4.28: Number of cycles and Cache miss ratio for memory move function of the CU for different data rates

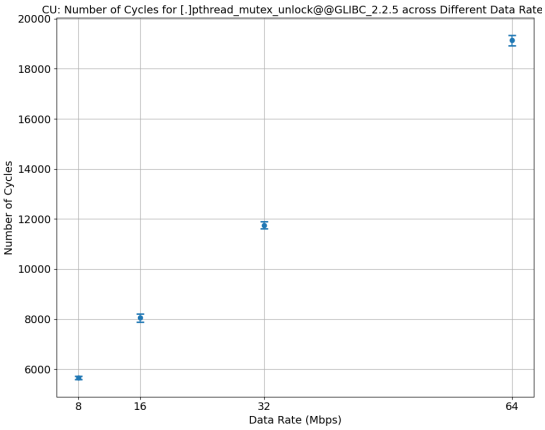


(a) Number of cycles for GTP function of the CU for different data rates

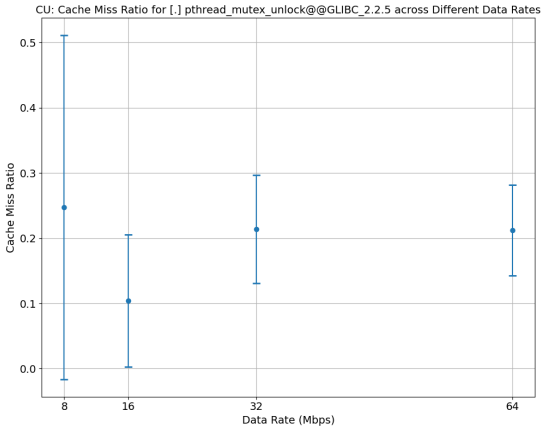


(b) Cache miss ratio for GTP function of the CU for different data rates

Figure 4.29: Number of cycles and Cache miss ratio for GTP function of the CU for different data rates



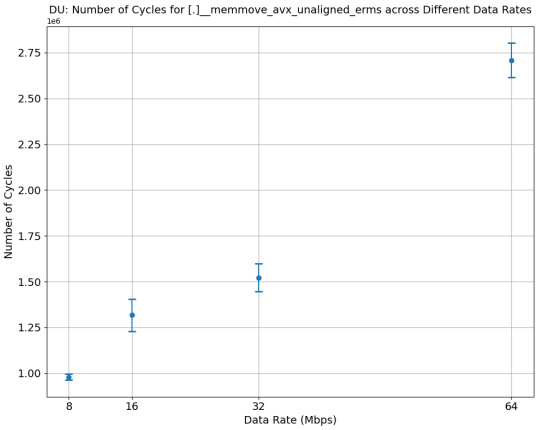
(a) Number of cycles for pthread mutex unlock function of the CU for different data rates



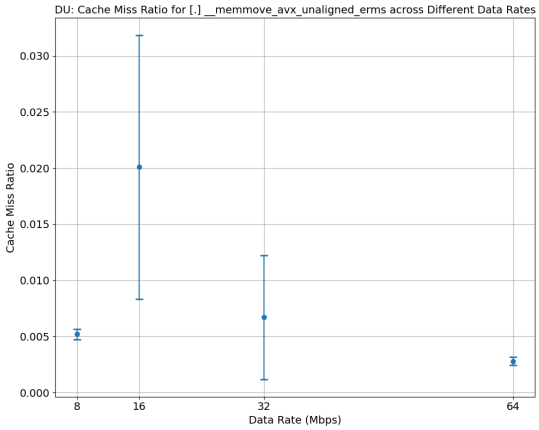
(b) Cache miss ratio for pthread mutex unlock function of the CU for different data rates

Figure 4.30: Number of cycles and Cache miss ratio for pthread mutex unlock function of the CU for different data rates

DU functions:

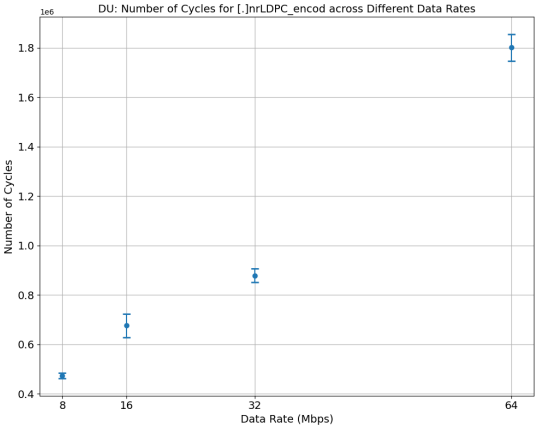


(a) Number of cycles for memory move function of the DU for different data rates

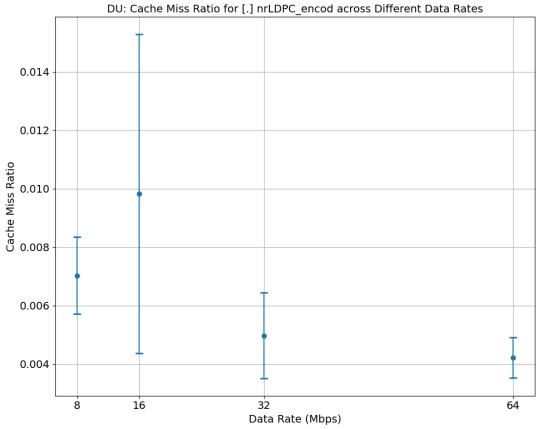


(b) Cache miss ratio for memory move function of the DU for different data rates

Figure 4.31: Number of cycles and Cache miss ratio for memory move function of the DU for different data rates



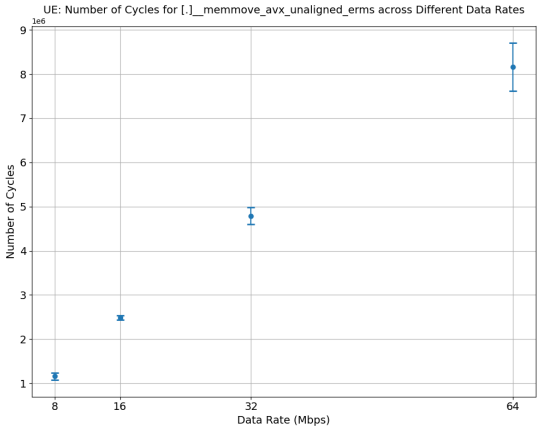
(a) Number of cycles for nrLDPC encoding function of the DU for different data rates



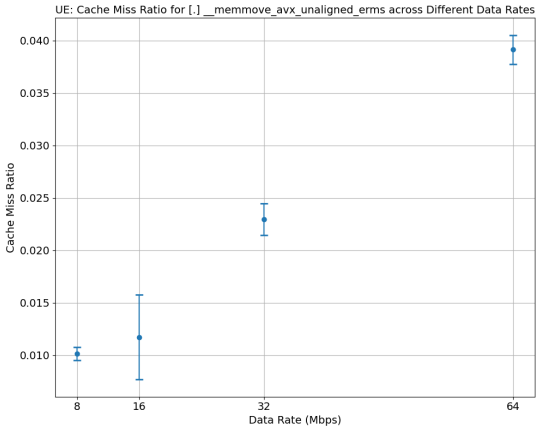
(b) Cache miss ratio for nrLDPC encoding function of the DU for different data rates

Figure 4.32: Number of cycles and Cache miss ratio for nrLDPC encoding function of the DU for different data rates

UE functions:

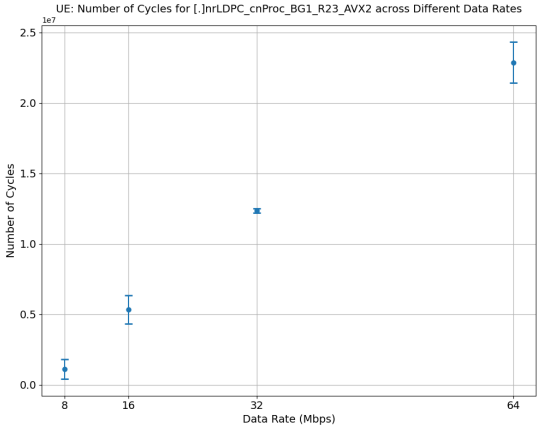


(a) Number of cycles for memory move of the UE for different data rates

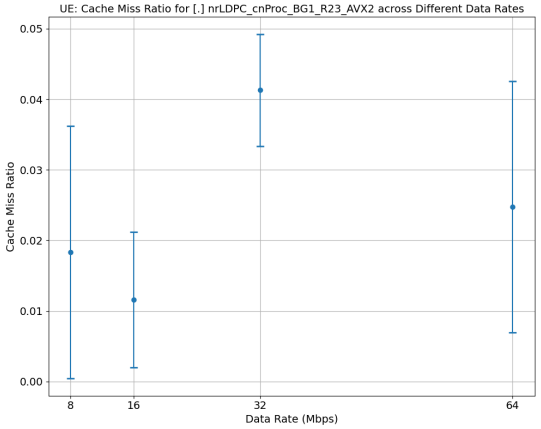


(b) Cache miss ratio for memory move function of the UE for different data rates

Figure 4.33: Number of cycles and Cache miss ratio for memory move function of the UE for different data rates

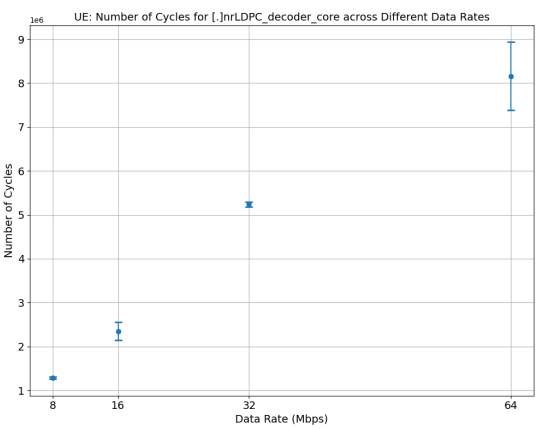


(a) Number of cycles for New Radio LDCP function of the UE for different data rates

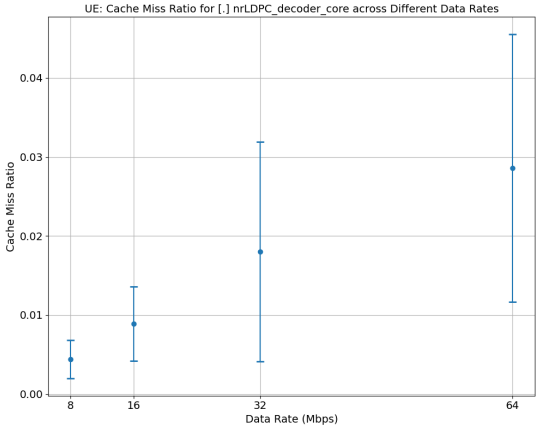


(b) Cache miss ratio for New Radio LDCP function of the UE for different data rates

Figure 4.34: Number of cycles and Cache miss ratio for New Radio LDCP function of the UE for different data rates



(a) Number of cycles for New Radio LDCP decoding function of the UE for different data rates



(b) Cache miss ratio for New Radio LDCP decoding function of the UE for different data rates

Figure 4.35: Number of cycles and Cache miss ratio for New Radio LDCP decoding function of the UE for different data rates

Chapter 5

Conclusion and Future work

In conclusion, this research has provided a comprehensive understanding of the problem at hand, focusing on the evaluation of 5G Non-Public Networks (NPN) performance in various scenarios, focusing on key dimensions like achievable throughput and power consumption. The analysis included determining the relationship between throughput and total power consumption, as well as the one between throughput and processor's power consumption.

Through this analysis, several conclusions have been drawn in alignment with the established and desired goals. It became clear that changes in throughput do not significantly impact the power consumption, either in terms of total power or processor power. Additionally, processor power consumption is not the sole contributor to total power consumption. Furthermore, despite deep dives into understanding which processor functions are influenced by throughput, the overall findings indicated negligible effects. However, certain functions were identified as potentially significant for optimizing in case of higher data rates, warranting attention. Another key finding was that the maximum achievable throughput fell short of initial expectations, with the achieved throughput being around 40-50 Mbps.

It is important to note that the results of this research might not apply to the networks having more than one gNB and more UE devices, since we only had one of each in our research. Nevertheless, the research gave valuable insights into the capabilities and limitations of 5G OAI framework, crucial for advancing various industries.

Looking ahead, ongoing advancements in the OpenAirInterface framework offer prospects for achieving higher throughput and potentially reducing power consumption in future iterations. Additionally, exploring the use of commercial User Equipment may offer enhanced solutions for the network's development. It would also be beneficial to perform the same analysis for uplink, and at some point use more than one UE.

References

[1] 3GPP. Non-public networks (npn). https://www.3gpp.org/technologies/npn, 2022. Accessed: 2024-06-25.

[2] 3rd Generation Partnership Project. Technical specification group radio access network; ng-ran; f1 general aspects and principles. 18.2.0(release 18):23, 2024.

[3] Maryam Amini and Catherine Rosenberg. A comparative analysis of open-source software in an e2e 5g standalone platform. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2024.

[4] Jose A Ayala-Romero, Ihtisham Khalid, Andres Garcia-Saavedra, Xavier Costa-Perez, and George Iosifidis. Experimental evaluation of power consumption in virtualized base stations. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[5] Wilfrid Azariah, Fransiscus Asisi Bimo, Chih-Wei Lin, Ray-Guang Cheng, Navid Nikaein, and Rittwik Jana. A survey on open radio access networks: Challenges, research directions, and open source approaches. *Sensors*, 24(3):1038, 2024.

[6] Arturo Bellin, Marco Centenaro, Nicola di Pietro, Arif Ishaq, Daniele Munaretto, Daniele Ronzani, Andrea Spinato, Stefano Tomasin, and Fabrizio Granelli. Autonomous private mobile networks: State of the art and future challenges. *IEEE Communications Standards Magazine*, 7(2):24–31, 2023.

[7] Chroma. Digital power meter 66203/66204 user's manual. https://www.chromausa.com/document-library/manuals-66203-66204/, 2016.

[8] Chroma. Digital power meter model 66203/66204. https://www.chromausa.com/product/multi-channel-digital-power-meters-66203-66204/, 2023.

[9] Jose Oscar Fajardo, Fidel Liberal, Ioannis Giannoulakis, Emmanouil Kafetzakis, Vincenzo Pii, Irena Trajkovska, Thomas Bohnert, Leonardo Goratti, Roberto Riggio, Javier Lloreda, Pouria Sayyad Khodashenas, Michele Paolino, Pavel Bliznakov, Jordi Pérez-Romero, Claudio Meani, Ioannis Chochliouros, and Maria Belesioti. Introducing mobile edge computing capabilities through distributed 5g cloud enabled small cells. *Mobile Networks and Applications*, 21, 08 2016.

[10] GeeksForGeeks. 5g network architecture. https://www.geeksforgeeks.org/5g-network-architecture/, 2022.

[11] Mohammad Asif Habibi, Meysam Nasimi, Bin Han, and Hans Schotten. A comprehensive survey of ran architectures toward 5g mobile communication system. *IEEE Access*, PP:1–51, 05 2019.

[12] Intel. Running average power limit energy reporting / cve-2020-8694 , cve-2020-8695 / intel-sa-00389. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html, 2022. Accessed: 2024-06-25.

[13] iPerf. iperf - the ultimate speed test tool for tcp, udp and sctp. https://iperf.fr/.

[14] iPerf. iperf - the ultimate speed test tool for tcp, udp and sctp. https://iperf.fr/iperf-doc.php.

[15] Sadiq Iqbal and Jehad M Hamamreh. A comprehensive tutorial on how to practically build and deploy 5g networks using open-source software and general-purpose, off-the-shelf hardware. *RS Open J. Innov. Commun. Tech*, 2(6):1–28, 2021.

[16] Tawfik Ismail and Haitham Mahmoud. Optimum functional splits for optimizing energy consumption in v-ran. *IEEE Access*, 8:194333 – 194341, 10 2020.

[17] ITU. Imt vision - framework and overall objectives of the future development of imt for 2020 and beyond. https://www.itu.int/rec/R-REC-M.2083, 2015.

[18] Rajeev Kumar, Andrea Francini, Shivendra Panwar, and Sameerkumar Sharma. Dynamic control of rlc buffer size for latency minimization in mobile ran. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2018.

[19] Lenovo. Legion pro 5 16irx9. https://psref.lenovo.com/Product/Legion/Legion_Pro_5_16IRX9, 2024. Accessed: 2024-06-25.

[20] OpenAirInterface. Oai 5g nr sa tutorial with oai nrue. https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/doc/NR_SA_Tutorial_OAI_nrUE.md, 2023. Accessed: 2024-06-25.

[21] Jorge Ortin, Pablo Serrano, Jaime Garcia-Reinoso, and Albert Banchs. Analysis of scaling policies for nfv providing 5g/6g reliability levels with fallible servers. *IEEE Transactions on Network and Service Management*, 19(2):1287–1305, 2022.

[22] Larry Peterson, Oguz Sunay, and Bruce Davie. Private 5g: A systems approach, 2022.

[23] Linh-An Phan, Dirk Pesch, Utz Roedig, and Cormac J. Sreenan. Building a 5g core network testbed: Open-source solutions, lessons learned, and research directions. In *2024 International Conference on Information Networking (ICOIN)*, pages 641–646, 2024.

[24] Ettus Research. Usrp b210. https://www.ettus.com/all-products/ub210-kit/, 2022.

[25] José Pedro Nogueira Rodrigues. Open source non-public 5g networking in licensed and unlicensed bands. 2023.

[26] Stefan Rommer, Peter Hedman, Magnus Olsson, Lars Frid, Shabnam Sultana, and Catherine Mulligan. Chapter 3 - architecture overview. In Stefan Rommer, Peter Hedman, Magnus Olsson, Lars Frid, Shabnam Sultana, and Catherine Mulligan, editors, *5G Core Networks*, pages 15–72. Academic Press, 2020.

[27] SUSE. Power management. https://doc.opensuse.org/documentation/leap/archive/42.2/tuning/html/book.sle.tuning/cha.tuning.power.html: :text=When Accessed: 2024-07-25.

[28] Wiki. Linux kernel profiling with perf. https://perf.wiki.kernel.org/index.php/Tutorial, 2024. Accessed: 2024-06-25.

Appendix A

Configuration files

A.1 Control Unit(CU)

cu_gnb.conf

```
    (...)
tracking_area_code = 7; #Chosen TAC
plmn_list = ({
    mcc = 208; #Chosen MCC
    mnc = 93; #Chosen MNC
    mnc_length = 2;
    snssaiList = ({ sst = 1 })
});
(...)
tr_s_preference = "f1"; #F1 interface, split between the CU and DU
local_s_if_name = "enp7s0"; #Network interface connecting CU and DU machine
local_s_address = "10.42.0.1"; #CU machine IP
remote_s_address = "10.42.0.2"; #DU machine IP
(...)
```

A.2 Distributed Unit(DU)

du_gnb.conf

```
    (...)
tracking_area_code = 7; #Chosen TAC
plmn_list = ({
    mcc = 208; #Chosen MCC
    mnc = 93; #Chosen MNC
    mnc_length = 2;
```

```
    snssaiList = ({ sst = 1 })
});
MACRLCs = ({
    num_cc = 1;
    tr_s_preference = "local_L1";
    tr_n_preference = "f1"; #F1 interface, split between the CU and DU
    local_n_if_name = "enp7s0"; #Interface connecting CU and DU machine
    local_n_address = "10.42.0.2"; #DU machine IP
    remote_n_address = "10.42.0.1"; #CU machine IP
    (...)
RUs = (
    (...)
    bands = [78]; #Chosen frequency band
    (...)
```

A.3 User Equipment (UE)

ue.conf

```
uicc0 = {
    imsi = "208930100001124";
    key = "465B5CE8B199B49FAA5F0A2EE238A6BC";
    opc= "E8ED289DEBA952E4283B54E88E6183CA";
    dnn= "internet";
    nssai_sst=1;
    imeisv="6754567890123413"
}
```