FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Autonomous Control and Positioning of a Mobile 5G Radio Access Node Employing the O-RAN Architecture

Gonçalo Moura Tomé Fraguito Queirós

Mestrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Manuel Ricardo

July 28, 2023

Abstract

Over the years, mobile networks were deployed using monolithic hardware based on proprietary solutions, where the network functions were part of a single closed unit. In 5G networks, this approach is still followed by Mobile Network Operators, due to the lack of suitable alternatives. Still, mobile networks have become more and more complex, especially to enable 5G. This complexity creates the need to produce, exchange and analyze high amounts of data, perform analytics, and configure several parameters across different network components. When those components rely on closed, proprietary solutions, this makes it difficult to deploy, configure, and optimize them. The concept of Open Radio Access Networks (RANs), including the standards and specifications proposed by the O-RAN Alliance, brings the possibility of creating open, interoperable networks based on independent virtualized components connected through standardised open interfaces. This paves the way for the deployment of RANs based on multi-vendor components, where the acquisition of metrics and the control of the RAN components can be performed by software applications such as the O-RAN-specified xApps. Another promising 5G research area is the usage of Mobile Robotic Platforms for the transportation of radio cells. These cells can be placed on demand to provide increased coverage, being especially useful in emergency scenarios and crowded events.

The main contribution of this dissertation is a private Standalone 5G Network with a mobile RAN employing the O-RAN architecture. The mobile RAN consists of a 5G radio node (gNB) carried by a Mobile Robotic Platform that is able to be autonomously positioned. The proposed solution employs a novel Mobility Management xApp, which is able to autonomously collect metrics from the RAN, analyze them, and use an original algorithm to determine and control the placement of the mobile RAN in order to increase the connection quality between the User Equipments (UEs) and the gNB. The proposed solution was experimentally validated considering a realistic use case.

Resumo

Ao longo dos anos, as redes móveis foram implementadas usando *hardware* monolítico baseado em soluções proprietárias, nas quais todas as funções de rede faziam parte de uma única unidade fechada. Nas redes 5G esta abordagem ainda é seguida pelos operadores de redes móveis, devido à falta de alternativas adequadas. No entanto, as redes móveis tornaram-se cada vez mais complexas, especialmente no que toca a novas tecnologias como o caso do 5G. Esta complexidade cria a necessidade de produzir, trocar e analisar grandes quantidades de dados, realizar análises, e configurar vários parâmetros em diferentes componentes da rede. Quando esses componentes dependem de soluções fechadas e proprietárias, isso dificulta a sua implementação, configuração e otimização. O conceito de redes de acess rádio (RANs) abertas, incluindo as normas e especificações propostas pela O-RAN *Alliance*, oferece a possibilidade de criar abertas e interoperáveis com base em componentes virtualizados, independentes e ligados através de interfaces abertas e normalizadas. Isto abre caminho à implementação de RANs baseadas em componentes de vários fornecedores, nas quais a aquisição de métricas e o controlo dos seus componentes podem ser realizados por aplicações de *software*, como as xApps especificadas pela O-RAN *Alliance*. Outra área de investigação promissora no domínio 5G consiste na utilização de plataformas robóticas móveis para o transporte de células rádio. Estas células podem ser posicionadas a pedido, por forma a proporcionar uma maior cobertura, sendo especialmente úteis em cenários de emergência e eventos lotados.

A principal contribuição desta dissertação é uma rede 5G *Standalone* privada com uma RAN formada por um nó móvel que usa a arquitetura O-RAN. A RAN móvel consiste num nó rádio 5G (gNB) transportado por uma plataforma robótica móvel capaz de se posicionar autonomamente. A solução proposta usa uma nova xApp de Gestão de Mobilidade, capaz de adquirir autonomamente métricas da RAN, analisá-las e utilizar um algoritmo original para determinar e controlar o posicionamento da RAN móvel, por forma a melhorar a qualidade da ligação entre os terminais (UEs) e o gNB. A solução proposta foi validada experimentalmente tendo em consideração um caso de uso realista.

Acknowledgments

I would like to thank my supervisors, Professor Manuel Ricardo and Dr. André Coelho, and also Eng. Paulo Furtado Correia for their support and concern during the development of this dissertation, and their guidance and assistance in the writing of this document.

I would also like to thank all the collaborators from the Center for Telecommunications and Multimedia (CTM) of INESC TEC for welcoming me and providing the necessary equipment and space for this dissertation's development. Their contributions have been crucial to its success.

Lastly, I must thank my friends and family, for their encouragement, and for providing moments of joy and relaxation throughout the process of developing this dissertation.

This work is co-financed by Component 5 - Capitalization and Business Innovation, integrated in the Resilience Dimension of the Recovery and Resilience Plan within the scope of the Recovery and Resilience Mechanism (MRR) of the European Union (EU), framed in the Next Generation EU, for the period 2021 - 2026, within project Produtech_R3, with reference 60.

Gonçalo Queirós

Contents

List of Figures

List of Tables

Abbreviations

3GPP	3rd Generation Partnership Project
AMF	Access and Mobility Management Function
AN	Access Network
API	Application Programming Interface
ARQ	Automatic Repeat reQuest
BBU	Base Band Unit
COTS	Commercial Off-the-shelf
CP	Cyclic Prefix
CPU	Central Processing Unit
C-RAN	Cloud-RAN
DL	Downlink
DN	Data Network
eCPRI	Enhanced Common Public Radio Interface
eMBB	Enhanced Mobile Broadband
eNB	eNodeB
ETSI	European Telecommunications Standards Institute
gNB	gNodeB
gNB-CU	gNB Central Unit
gNB-CU-CP	gNB Central Unit-Control Plane
gNB-CU-UP	gNB Central Unit-User Plane
gNB-DU	gNB Distributed Unit
GPRS	General Packet Radio Service
GTP-U	GPRS Tunnelling Protocol for user data
HARQ	Hybrid ARQ
HTTP	Hypertext Transfer Protocol
I/Q	In-phase and Quadrature
IAB	Integrated Access and Backhaul
ICMP	Internet Control Message Protocol
iFFT	Inverse Fast Fourier Transform
IMSI	International Mobile Subscriber Identifier
IMT-2020	International Mobile Telecommunications-2020
IMT-Advanced	International Mobile Telecommunications-Advanced
IP	Internet Protocol
ITU	International Telecommunication Union
ITU-R	ITU Radiocommunication Sector
ITU-T	ITU Telecommunication Standardization Sector

KPM	Key Performance Metrics
LED	Light-Emitting Diode
MAC	Medium Access Control
MCC	Mobile Country Code
MCI	Mobility Control Interface
ML	Machine Learning
mMTC	Massive Machine Type Communications
MNC	Mobile Network Code
NAS	Non-Access-Stratum
NAT	Network Address Translation
Near-RT RIC	Near-Real Time RAN Intelligent Controller
NETCONF	Network Configuration Protocol
NFV	Network Functions Virtualization
NGAP	NG Application Protocol
Non-RT RIC	Non-Real Time RAN Intelligent Controller
NR	New Radio
NR-Uu	5G Radio Interface
NRF	Network Repository Function
NUC	Next Unit of Computing
OAI	Open Air Interface
O-CU-CP	O-RAN Central Unit–Control Plane
O-CU-UP	O-RAN Central Unit–User Plane
O-DU	O-RAN Distributed Unit
O-RU	O-RAN Radio Unit
OPc	Operator Code
OS	Operating System
OSC	O-RAN Software Community
OSI	Open Systems Interconnection
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
PFCP	Packet Forwarding Control Protocol
PHY	Physical Layer
PLMN	Public Land Mobile Network
PTP	Precision Time Protocol
QoE	Quality of Experience
QoS	Quality of Service
RACH	Random Access Channel
RAM	Random-Access Memory
RAN	Radio Access Network
RF	Radio Frequency
RLC	Radio Link Control
RoE	Radio over Ethernet
RRC	Radio Resource Control
RRU	Remote Radio Unit
SA	Standalone
SCTP	Stream Control Transmission Protocol
SDAP	Service Data Application Protocol
SDK	Software Development Kit
SDN	Software Defined Networking
SDR	Software-Defined Radio

SDU	Service Data Unit
SFF	Small Form Factor
SIM	Subscriber Identity Module
SMF	Session Management Function
SMO	Service Management and Orchestration
SNR	Signal-to-Noise Ratio
SSH	Secure Shell Protocol
SyncE	Synchronous Ethernet
TAC	Tracking Area Code
TLS	Transport Layer Security
UDP	User Datagram Protocol
UE	User Equipment
UHD	USRP Hardware Driver
UL	Uplink
UPF	User Plane Function
URLLC	Ultra-Reliable and Low Latency Communications
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
VM	Virtual Machine
VRAN	Virtual RAN

Chapter 1

Introduction

1.1 Context

Mobile networks are constantly evolving in order to improve wireless communications and meet the ever-increasing Quality of Service (QoS) requirements of users, including increased throughput, lower latency, and higher availability. Throughout the multiple generations of mobile networks, new technologies have been introduced in order to meet those requirements, but the networks have become increasingly complex. Such complexity creates the need to properly configure, manage, and optimize the networks, which implies the exchange of data, analytics, and control messages between different components of the network.

The need to exchange and analyze high amounts of data and configure parameters across the network motivates the interfaces between the different network components to be open and available for configuration. However, following the paradigm of previous generations of mobile networks, the recently released commercial implementations employing the fifth-generation (5G) New Radio (NR) technology mainly rely on proprietary hardware solutions, especially at the radio access network (RAN) level.

When it comes to 5G network implementations, a paradigm that has arisen over the last few years is the concept of on-demand wireless connectivity. This can be achieved by means of integrating mobile robotic platforms acting as mobile access points in the network, able to be dynamically positioned on-demand, in order to provide better coverage and increase network performance.

1.2 Motivation and Problem

Solutions implementing the RAN are typically all-in-one units, in which all the layers of the protocol stack are seen as *black boxes* by network operators. This leads to a plethora of challenges, including: 1) vendor lock-in limitations, as solutions from a given vendor may be not compatible with solutions from other vendors; 2) restricted configuration options, which difficult fine-tuning several parameters within the devices; 3) limited inter-node coordination, which prevents joint

control of the RAN components; and 4) increased unit price, since the network equipment market is relatively small compared with, for example, industry-grade server market. This makes the number of units sold also small, and leads to expensive solutions in order to cover their development costs. This reality brings multiple challenges when it comes to deploying a 5G network that needs to be configured in order to meet specific requirements associated with different use cases.

In order to address these limitations, in 2018, the O-RAN Alliance was founded [16]. O-RAN Alliance is a community that joins mobile operators, vendors, and academic and research institutions with the main goal of leading the industry towards open, interoperable, and virtualized RAN components, by specifying a set of Application Programming Interfaces (APIs), interfaces, and standards. O-RAN paves the way for mobile network operators to deploy networks tailored to the performance requirements of specific use cases, while minimizing the dependence on proprietary hardware and enabling the implementation of networks whose components can be acquired from different vendors.

From the network performance point of view, mobile networks should meet the users' requirements, typically defined by means of target QoS levels. Achieving this in some scenarios may be challenging to fixed RANs, including in crowded events, where the obstruction to the line of sight between the user equipment (UE) and the radio cells' antennas can cause significant degradation of the QoS offered, due to network congestion and wireless connectivity failures. Placing a radio cell (gNodeB) on a mobile robotic platform, which can be repositioned on demand, paves the way to provide enhanced wireless channel conditions and improved QoS to the UE. However, such implementation design comes with some problems, namely: 1) how can we assess the network conditions and determine the best placement of the mobile gNodeB; and 2) how can we incorporate this design paradigm into the existing network architectures.

1.3 Objectives

The main goal of this dissertation was to design, develop, and evaluate a private 5G Standalone (SA) RAN, based on the O-RAN architecture, composed of a RAN node deployed on a mobile robotic platform. In order to achieve this, the following specific objectives were defined:

- Specify and design a private 5G SA RAN, based on the O-RAN architecture, consisting of a mobile RAN node.
- Develop an algorithm, implemented as a Mobility Management xApp, which is able to retrieve metrics from the RAN and use them to determine better placements for the mobile RAN node.
- Implement a testbed of the 5G O-RAN based network designed, including the mobile RAN node and the Mobility Management xApp.
- Test and validate the designed network and collect performance results.

1.4 Contributions

The main contributions of this dissertation are three-fold:

- A private 5G SA mobile RAN, based on the O-RAN architecture, controlled by a mobility management function able to monitor the network and dynamically position the mobile RAN node for increased performance.
- A Mobility Management xApp capable of autonomous control and placement of the mobile RAN, in order to increase the 5G connection quality.
- An O-RAN based testbed for research on emerging networks, which can be used to test state of the art mobility management functions in multiple scenarios.

1.5 Document Structure

This document is structured as follows. In Chapter 2, we present the state of the art and related work, including the fundamental concepts related to the problem of this dissertation. In Chapter 3, we introduce the proposed solution, detailing the system specification, design choices, and how it was implemented. In Chapter 4, we present the tests that were conducted in order to validate and evaluate the proposed solution and its implementation; finally. In Chapter 5, we draw the main conclusions and present future work.

Chapter 2

State of the Art

In this chapter, we present fundamental concepts and existing solutions related to the problem of this dissertation. In Section 2.1, we present an introduction to 5G, addressing its performance targets and new technologies introduced while comparing it to previous generations of mobile networks. In Section 2.2, the main components and interfaces of the 5G Architecture are presented and explained. In Section 2.3, we explore different RAN deployment approaches, including O-RAN, making comparisons between them. In Section 2.4, the O-RAN architecture and specifications are presented. Finally, in Section 2.5, different solutions for implementing O-RAN compliant 5G network deployments are described.

2.1 5G Characterization

Generation after generation, the network requirements and the capabilities to meet such requirements have been growing. When it comes to the fifth-generation (5G) mobile networks, significant enhancements compared to the fourth-generation (4G) networks have been defined and quantified.

In 2015, the International Telecommunication Union (ITU) Radiocommunication Sector (ITU-R) published the Recommendation ITU-R M.2083-0 [1], where the initial targets for research and investigation into the 5th generation mobile communications are set. Those targets include key capabilities that should be achieved by the 5G technology, such as latency, user-experienced data rate and connection density. Figure 2.1 depicts the enhancements from International Mobile Telecommunications-Advanced (IMT-Advanced) (4G requirements) to IMT-2020 (5G requirements).

Besides the enhancements of these capabilities over 4G, 5G have introduced the concept of network slicing. In previous generation networks, like 4G, an "one-size-fits-all" design is followed when it comes to the network resources and capabilities offered for each User Equipment (UE). However, if every UE is provided with all the capabilities of the network, such as those shown in Figure 2.1 for 5G, this may imply wasting network resources, because there are use cases that do not take advantage of all these capabilities simultaneously. Network slicing consists in partitioning a single physical shared network infrastructure into multiple virtual networks, named

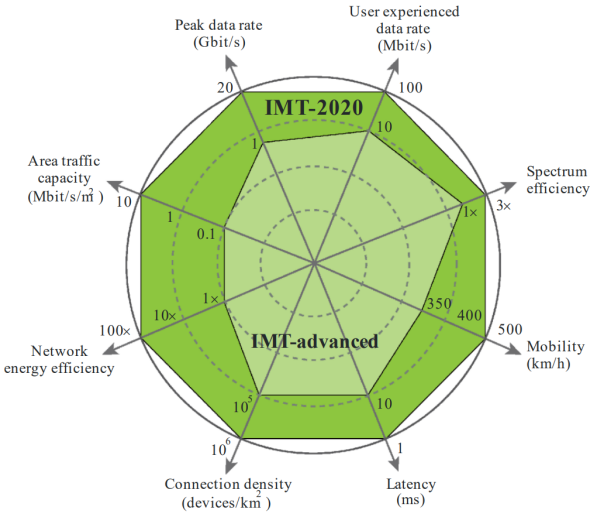

Figure 2.1: Enhancement of key capabilities from IMT-Advanced (in light green) to IMT-2020 (in dark green) [1].

network slices. Each virtual network is designed and optimized for a specific type of application or communications service [17]. This is achieved by means of virtualization approaches which allow us to dynamically allocate and schedule the resources of the physical network to be used by different virtual networks. An analogy of the network slicing concept is to instantiate multiple Virtual Machines (VMs) (i.e., the network slices) in a single host Operating System (OS) (i.e., the 5G network) running on a single computer (i.e., the physical shared network infrastructure). Network slices are self-contained networks logically separated from each other, with their own traffic flows, traffic rules and network topologies. From the end-user perspective, each network slice virtually uses a dedicated network; in practice, multiple network slices transparently share the same physical hardware.

In 5G, there are three main classes of use case scenarios: Enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), and Massive Machine Type Communications (mMTC). They are typically met by different network slices, where each use case takes advantage of different key capabilities that allow them to fulfil their needs when it comes to network performance. The importance of each key capability for these three use case scenarios, highlighted by different colours, is depicted in Figure 2.2. The three overall use cases are described as follows:

- **eMBB**: It represents an evolution of the traditional mobile-broadband services available in previous generation networks such as 4G. eMBB is focused on enabling larger data transfer capabilities and enhanced user Quality of Experience (QoE). It also aims at providing a seamless experience when it comes to wireless coverage, including in dynamic scenarios, by ensuring not only high peak but also constant data rates in the covered areas, while taking into account the requirements imposed by high-speed transportation (e.g., high-speed trains).

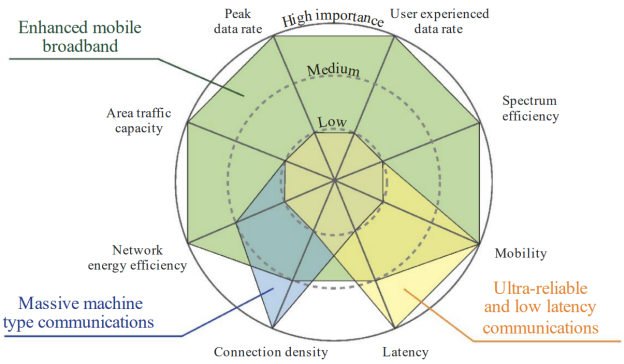

Figure 2.2: The importance of key capabilities in different scenarios [1].

- **URLLC**: This category is focused on communication services and applications that are time critical and need ultra-reliable network connectivity, demanding strict requirements when it comes to latency and network availability. Reference examples include traffic safety, automatic control, remote medical surgery, autonomous driving, wireless control of industrial manufacturing, and production services.
- **mMTC**: This use case is characterized by communication services and applications for a massive number of connected devices transmitting low amounts of non-delay sensitive data. These devices are typically characterized by their low cost and the need to be energy efficient, such as remote sensors and actuators massively deployed per unit of area in smart city environments.

Even though these are the three main categories of use cases envisioned by 5G, they should be seen as high-level guidelines for network design purposes; there are some scenarios and applications whose network requirements may not fit exactly into one of these three categories. For example, some applications may require high network reliability and availability, but may not require low-latency communication. Figure 2.3 depicts specific use case scenarios and where they fit in the three main categories of 5G use cases.

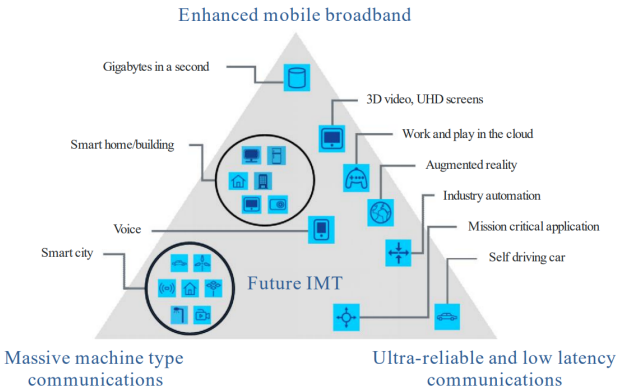

Figure 2.3: Use case scenarios associated with the three main types of network slices [1].

2.2 5G System Architecture

The responsibility of specifying the technologies that allow meeting the requirements that were explored in Section 2.1 belongs to the 3rd Generation Partnership Project (3GPP). 3GPP joins several telecommunications standard development organizations, whose main goal is to develop and provide mobile broadband standards and technical specifications [18].

Regarding 5G, 3GPP has standardised the 5G New Radio (NR) architecture, by developing and providing technical specifications for both the 5G Radio Acess Network (RAN) and the 5G Core Network (CN).

When it comes to the 5G NR architecture, it follows the principles of Software Defined Networking (SDN). SDN is a network architecture where the network functions are divided into two planes: the control plane and the data/forwarding plane [19]. This architecture allows for independent scalability and flexible deployments. A high-level reference architecture of 5G is depicted in Figure 2.4.

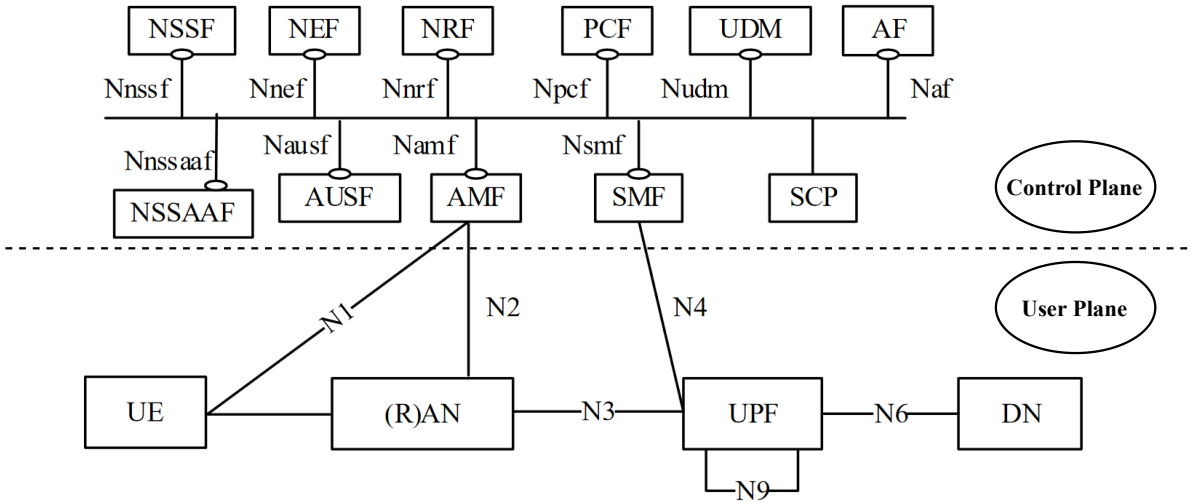

Figure 2.4: 5G reference architecture, adapted from [2].

In the case of the 5G Architecture, the main components of the data plane, also known as the user plane, the UE, the RAN, and the User Plane Function (UPF). They are connected to the Access and Mobility Management Function (AMF) and the Session Management Function (SMF), which belong to the control plane and are part of the Core Network in the context of 5G. The data plane is responsible for forwarding data packets to/from the UE.

The RAN is implemented by means of a base station, also called gNodeB (gNB) in the context of 5G. It ensures wireless connectivity between the UE and the 5G network and is responsible for network functions involving radio-related communications. Besides allowing radio transmissions of user data packets, it is also connected to the CN, creating a control link between the UE to the 5G CN.

The UPF acts as the default gateway for user data packets forwarded to/from the UE, connecting the network to the Data Network (DN), for example, the Internet. It also acts as an anchor point for data flows during Intra-/Inter-RAT mobility, while ensuring the data flows are properly forwarded when the UE connects to a different RAN in the same network. Moreover, the UPF defines rules, handles and monitors the QoS levels offered by the user plane, and reports traffic usage.

The control plane is responsible for handling the signalling and control packets and providing services to mobile subscribers. Even though Figure 2.4 depicts the multiple network functions associated with the control plane, herein we give special emphasis to AMF and SMF, as they are in charge of connecting the RAN to the Core Network.

The AMF's main roles are to establish and manage the 5G control connection between the CN and the UEs. When a UE first connects to the network, the AMF is in charge of verifying if the UE is registered in the network and is allowed to use the network's services. The AMF also acts as the gateway for the control traffic exchanged between the Acess Network (AN) and the UE, while forwarding control and signalling messages to/from the remaining 5G core network functions, such as the SMF. Besides this, mobility events, such as handovers between gNBs, are managed by the AMF.

The SMF is in charge of managing the data sessions between the UPF and the AN, being responsible for establishing, modifying, and releasing those sessions. It is also responsible for the allocation and management of IP addresses for the UE; the IP addresses can be received from the UPF or an external DN authorized by the SMF. The SMF is also responsible for other IP-related functions such as implementing Dynamic Host Configuration Protocol (DHCP).

The 5G NR radio protocol stack is used to exchange packets or Protocol Data Units (PDUs) between the UE and the RAN, via the radio interface between the two, denominated NR-Uu interface in the context of 5G. This protocol stack is similar to the 4G protocol stack when it comes to the sublayers and protocols used [20]. It has two different configurations: one for exchanging user plane PDUs and another for exchanging control plane PDUs. The sublayers of the 5G NR protocol stack for the user plane are depicted in Figure 2.5. The different sublayers and their functions for the user plane protocol stack are the following:

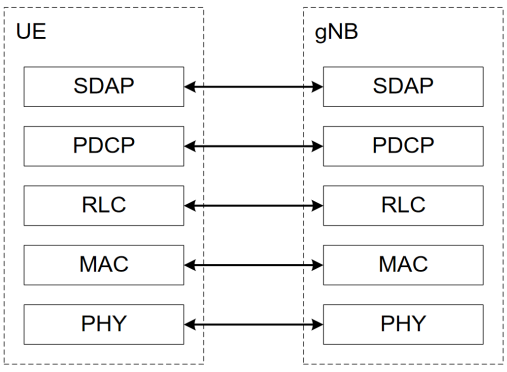

Figure 2.5: 5G NR User Plane Protocol Stack [3].

- **Service Data Application Protocol (SDAP)**: This sublayer functions include mapping the QoS flow to the respective radio bearer and marking packets with a QoS flow ID (QFI). Every individual PDU session has a single SDAP protocol entity configured and assigned to it.
- **Packet Data Convergence Protocol (PDCP)**: PDCP's main functions are header compression and decompression, ciphering, and integrity verification. It also allows performing duplicate discarding, and both out-of-order delivery and reordering, ensuring in-order delivery.
- **Radio Link Control (RLC)**: RLC enables segmentation of PDCP PDUs, detection of duplicates, and error correction and retransmission using Automatic Repeat reQuest (ARQ).
- **Medium Access Control (MAC)**: This sublayer performs mapping between logical channels and transport channels, multiplexing of MAC Service Data Units (SDUs) between logical channels and transport blocks delivered to the physical layer, error correction and retransmission using Hybrid ARQ (HARQ) retransmissions, and priority handling between UEs and logical channels of the same UE, using scheduling and logical channel prioritisation, respectively.
- **Physical Layer (PHY)**: This is the lowest layer on the 5G NR protocol stack and it performs coding, rate matching, scrambling, modulation, layer mapping, and multi-antenna mapping.

Regarding the 5G NR control plane protocol stack, one of the main differences when compared with the 5G NR user plane protocol stack, is the replacement of the SDAP protocol by the Radio Resource Control (RRC) protocol, as depicted in Figure 2.6.

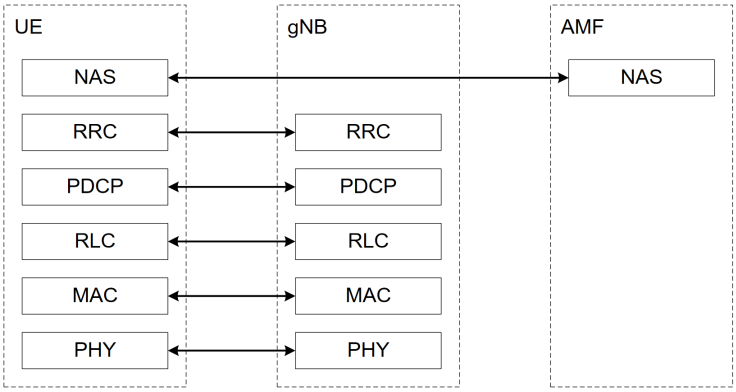

Figure 2.6: 5G NR Control Plane Protocol Stack [3].

RRC's main functions include the broadcast of system information, key management and security, mobility management, such as handovers, QoS control, and UE measurement reporting.

As depicted in Figure 2.6, the 5G NR control plane protocol stack also introduces the Non-Access-Stratum (NAS). This layer creates the N1 interface, which ensures connectivity between

the UE and the AMF of the 5G CN (cf. Figure 2.4). NAS is responsible for managing and transmitting signalling and control messages between the UEs and the AMF, supporting mobility functions, authentication and, identification. It is also responsible for handling and transmitting messages related to the establishment and management of data connectivity to external networks[21]. The N1 interface acts as a logical interface for the signalling messages exchanged between the UE and the AMF. The wireless connection between the UE and the RAN is ensured by the 5G NR protocol stack.

The connection between the RAN and AMF is ensured via the N2 interface (cf. Figure 2.4). The N2 interface, also known as the NG-c interface, uses the NG Application Protocol (NGAP), in order to support both (1) non UE-related services, associated to the N2 interface connection between the AMF and the RAN, and (2) UE-related services, for handling and forwarding signalling messages related with mobility, authentication, and identification, from the N1 interface and the NAS layer [22]. The delivery of the NGAP messages is ensured by the Stream Control Transmission Protocol (SCTP), which utilises multihoming and establishes redundant connections, increasing reliability. The protocol stack according to the Open Systems Interconnection (OSI) model for the N2 interface is depicted in Figure 2.7.

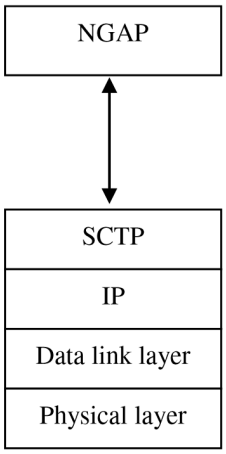

Figure 2.7: N2 Protocol Stack [4].

For the UE to communicate with the UPF and reach the DN, the UPF is in charge of forwarding data packets to/from the RAN using the N3 interface, also known as the NG-u interface. This interface provides non-guaranteed delivery of PDU Session user plane data between the RAN and the UPF. N3 implements the General Packet Radio Service (GPRS) Tunnelling Protocol for user data (GTP-U), which creates GTP-U Tunnels between the RAN and UPF, each carrying encapsulated PDUs belonging to a given data flow [23]. This allows packets to be multiplexed and de-multiplexed according to the data flow they belong to. The protocol stack for the N3 interface is depicted in Figure 2.8.

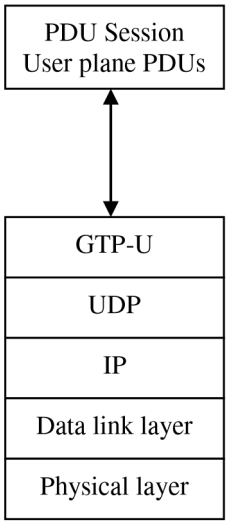

Figure 2.8: N3 Protocol Stack [4].

The N4 interface handles communication between the SMF and UPF. This means that this interface is responsible for interconnecting a core function that belongs to the Control Plane, in this case, the SMF, with a core function that belongs to the User Plane, in this case, the UPF, as depicted in Figure 2.9. As is the case with the N3 interface, the N4 interface also relies on UDP, but the application protocol that handles signalling and control procedures is the Packet Forwarding Control Protocol (PFCP). PFCP is used to establish a connection between SMF and UPF, to ensure the initial configuration of the UPF with rules, such as bandwidth limitations for different UEs or for different traffic flows, to send reports to the SMF for new traffic flows and to send heartbeat requests and responses in order to find if the peer PFCP entity is alive[5].

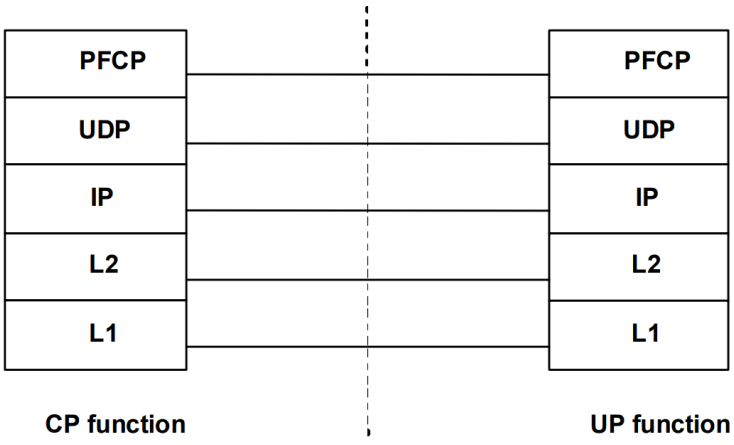

Figure 2.9: N4 Protocol Stack, between a Control Plane function (e.g., SMF) and a User Plane Function (e.g., UPF) [5].

2.3 RAN Deployment Approaches

Network equipment providers typically do not ensure interoperability between their equipment and other providers' equipment when designing RAN implementations relying on closed systems. Because of that, network operators need to acquire RAN solutions from a single provider. When it comes to implementing the RAN at a base station, it is usually divided into the Base Band Unit (BBU), which manages the base station and ensures connectivity with the Core network, and the Remote Radio Unit (RRU), which is connected to the base station's antenna. When these components are closed systems, they use closed interfaces, which may employ proprietary implementations defined by the network equipment provider. This approach is depicted in Figure 2.10.

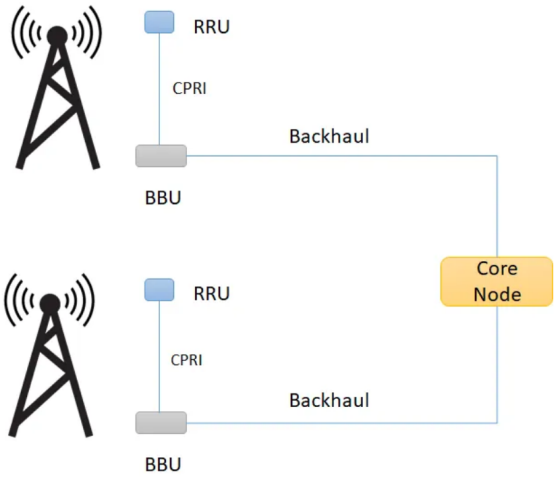

Figure 2.10: Traditional RAN approach [6].

Cloud-RAN (C-RAN) is a different type of approach for deploying the RAN. Instead of using a BBU at every base station, C-RAN centralizes most of the baseband processing for the RAN in cloud data centers, which connect to the RRU in the base stations using Fronthaul interfaces. In these cloud centers, the BBUs are implemented as virtualized nodes aggregated in a pool running on a single machine. This allows for capacity load balancing and enhanced signal processing capabilities. Still, this approach does not solve the vendor lock-in problem when the systems and interfaces rely on proprietary implementations.

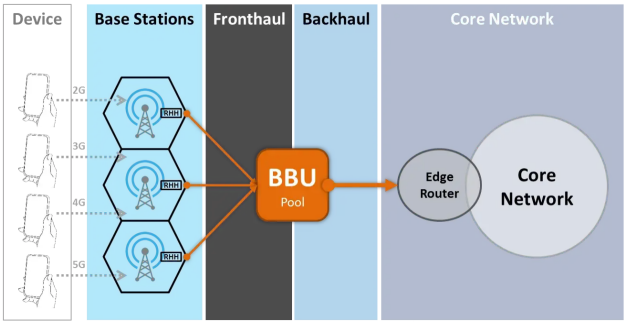

Figure 2.11: C-RAN deployment approach [7].

Another approach for deploying the RAN functions is Virtual RAN (VRAN). In C-RAN, even though the BBUs are virtualized, they run on top of proprietary hardware. VRAN's approach consists in replacing the proprietary hardware with Commercial Off-the-shelf (COTS) hardware, on top of which the BBU nodes can be deployed. VRAN partially solves the vendor lock-in limitations, because it allows running any vendor's virtualized network functions on any vendor's COTS hardware. However, the interfaces between the RRU and the virtualized BBUs are typically closed; as such, the virtualized software for the BBU and the RRU may not guarantee interoperability between solutions from different vendors.

To address this, an alternative deployment option was standardized by 3GPP. It consists in disaggregating the gNB, into a gNB Central Unit (gNB-CU) which is connected to one or more gNB Distributed Units (gNB-DUs) by the F1 interface, as depicted in Figure 2.12.

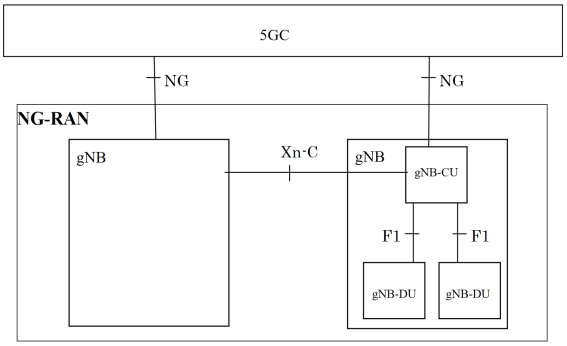

Figure 2.12: gNB disaggregation into gNB-CU and gNB-DU [8].

The gNB-CU can be further split in order to separate the Control Plane functions and the User Plane functions between two separate components, the gNB-CU-Control Plane (gNB-CU-CP) and the gNB-CU-User Plane (gNB-CU-UP), which are connected by the E1 interface, as depicted in Figure 2.13.

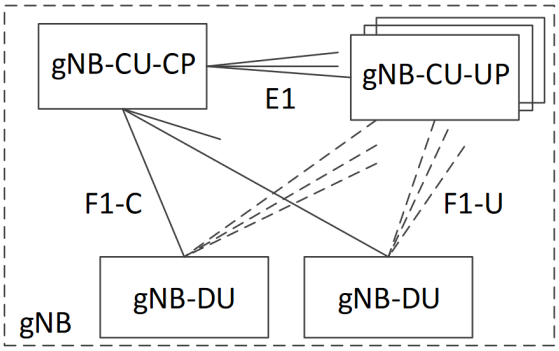

Figure 2.13: Separation of gNB-CU into gNB-CU-CP and gNB-CU-UP [8].

The E1 and F1 (F1-c and F1-u, in case of gNB-CU separation) interfaces are open, allowing connection between gNB-CUs and gNB-DUs from different vendors. This approach helps to mitigate the vendor lock-in problem.

The Open Radio Access Network (O-RAN) approach is the most recent paradigm paving the way for RAN open deployments. As VRAN, O-RAN relies on virtual network functions running in COTS hardware. O-RAN also builds upon the 3GPP disaggregation of the gNB, while specifying further RAN disaggregation and standardizing new open interfaces. Following this approach, it is possible to deploy a RAN composed of different vendors' network functions interacting between themselves using standardized open interfaces. The O-RAN architecture is the main focus of this dissertation. We give special emphasis to O-RAN in the following section.

2.4 O-RAN

Traditional 5G RANs typically consist in closed monolithic units implementing all network functions, including all the layers of the 5G NR protocol stack and all network interfaces. This may lead to multiple challenges, such as vendor lock-in limitations, restricted configuration options, increased unit price, and limited inter-node coordination. In order to change this approach, the O-RAN Alliance was founded. O-RAN focuses on reshaping the Radio Access Networks to be more intelligent, open, virtualized, and fully interoperable [24]. To implement the concept of an open RAN, one of O-RAN Alliance's main goals is to promote V-RAN components, considering the European Telecommunications Standards Institute (ETSI) concept for Network Functions Virtualization (NFV) [25]. According to the NFV concept, instead of using tailored proprietary hardware solutions to implement a RAN communications node, COTS hardware is used, on top of which the network functions are implemented by means of software [26]. Such network functions can be provided in the form of a virtual environment running on the Host Operating System, in the form of Virtual Machines or Containers. This allows for minimizing dependence on proprietary hardware.

By relying on independent RAN components and standardised interfaces between them, while benefiting from the synergy between hardware and virtualized network functions, O-RAN based

networks may be deployed by running any vendor's software on any vendor's hardware, as well as employing a combination of solutions from different vendors. This allows for the deployment of RANs tailored to specific use cases and enables new players to make part of the mobile networks market, since software solutions running on top of COTS servers represent an easier and cheaper approach to develop new carrier-grade products.

Regarding the O-RAN architecture and standards, the O-RAN Alliance builds upon the 3GPP disaggregation of the gNB presented in Section 2.3. Figure 2.14 depicts the logical architecture of O-RAN, including its components and interfaces; part of them were standardised by 3GPP and others were standardised by the O-RAN Alliance. The most important components and their functions include:

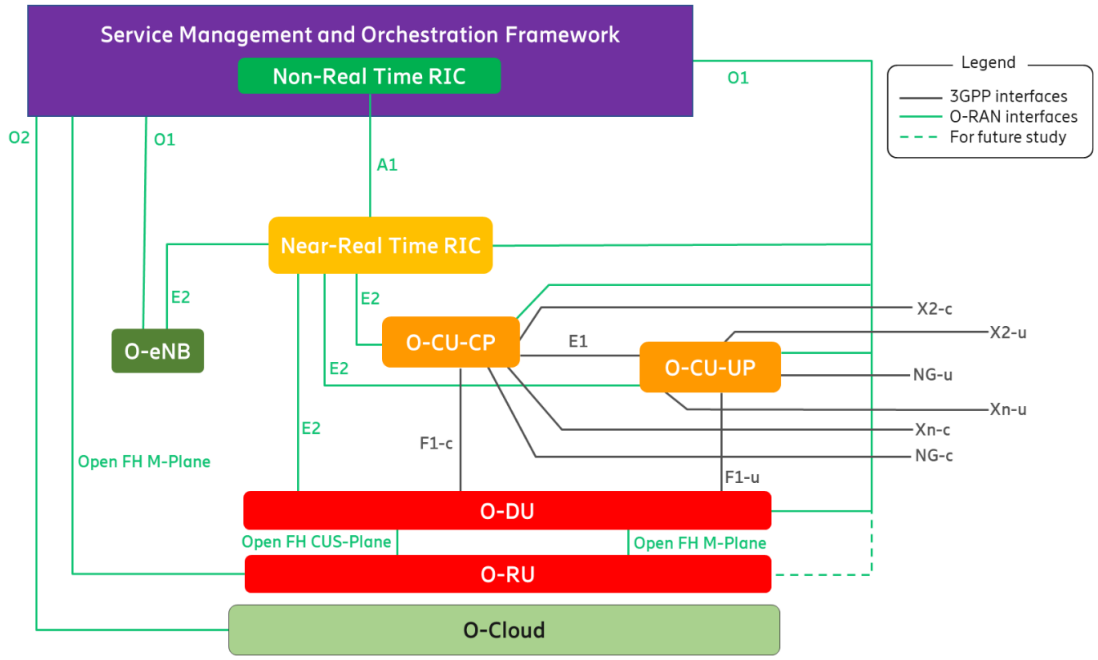

Figure 2.14: Logical Architecture of O-RAN [9].

- **O-RAN Central Unit–Control Plane (O-CU-CP)**: This component implements the 3GPP gNB-CU-CP. It connects the RAN to the AMF through the N2/NG-c interface and implements the higher layers of the 5G NR protocol stack for the control plane, namely the RRC and PDCP layers.

- **O-RAN Central Unit–User Plane (O-CU-UP)**: This component implements the 3GPP gNB-CU-UP. It connects the RAN to the UPF through the N3/NG-u interface and implements the higher layers of the 5G NR protocol stack for the user plane, namely the SDAP and PDCP layers.

- **O-RAN Distributed Unit (O-DU)**: This component implements the 3GPP gNB-DU. It is responsible for hosting the RLC and MAC layers of the 5G NR protocol stack. When

compared to its 3GPP counterpart, O-DU performs the upper functionalities of the Physical Layer (High-PHY), such as coding, rate matching, scrambling, modulation and layer mapping.

- **O-RAN Radio Unit (O-RU)**: O-RU was introduced by O-RAN architecture to enable some of the functions performed by the 3GPP gNB-DU. This component implements the lower functionalities of the Physical Layer (Low-PHY), such as precoding, remapping, digital beamforming, Inverse Fast Fourier Transform (IFFT), and Cyclic Prefix (CP) insertion. O-RU is typically connected to an antenna array to transmit and receive the modulated signal.

- **Near-Real Time RAN Intelligent Controller (Near-RT RIC)**. Contrary to the previous components, which were mostly derived from 3GPP standards, the RICs are defined by the O-RAN Alliance. The main function of the Near-RT RIC is to control and optimize the RAN. For that purpose, it provides a suitable environment for running multiple *xApps*. An *xApp* is a piece of software that runs in the Near-RT RIC in order to control and fine-tune the remaining O-RAN components in near-real time (up to 1 s). It collects data from the RAN (e.g., Key Performance Metrics (KPMs) such as packet loss ratio or radio resource utilization), while analyzing it and triggering control actions if necessary. The Near-RT RIC can also run pre-trained Machine Learning (ML) models to better configure the RAN.

- **Non-Real Time RAN Intelligent Controller (Non-RT RIC)**: This component is part of the Service Management and Orchestration (SMO) framework. It complements the near-RT RIC by providing additional information, managing ML models, and setting overall policies in non-real time (more than 1 s). Similarly to the near-RT RIC, the non-RT RIC also hosts third-party applications (*rApps*) for controlling the RAN. The *rApps* are focused on providing high-level policies, which manage the RAN as a whole, for example, providing the RAN initial setup. The non-RT RIC is able to control and manage multiple RANs.

The split employed by the O-RAN Alliance is Option 7.2x, which divides the PHY functions between the O-DU and O-RU. A representation of this split is depicted in Figure 2.15. This split option was chosen because it allows achieving a balance between the simplicity of the O-RU, and the data rates and latency required for the interface between the O-DU and O-RU [27]. Despite the fact that split option 7.2x is the default split for the O-RAN architecture, the O-RAN Alliance also considers the possibility of using split option 6, for which all PHY functions are performed by the O-RU, and split option 8, which considers that all PHY functions are performed in O-DU, while the O-RU is only in charge of Radio Frequency (RF) to baseband conversion [28]. Split options 6 and 8 are not yet fully supported and incorporated by O-RAN.

Some of the O-RAN interfaces are adopted from the 3GPP standards. Xn-c and Xn-u are used to connect the RAN to other gNBs in the control plane and the user plane, respectively. X2-c and Xn-u are used to connect the RAN to other eNodeBs (eNBs), the radio base stations in 4G networks, in the control plane and the user plane, respectively. The F1-c interface is used to interconnect the O-CU-CP and the O-DU, while managing the transmission of control plane

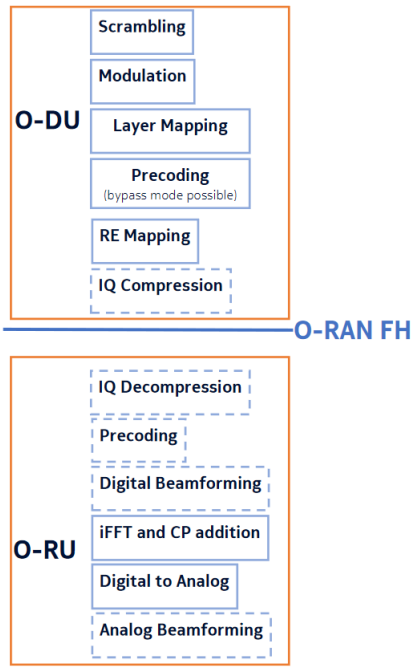

Figure 2.15: Physical layer functions separation between O-DU and O-RU, considering split option 7.2x [10].

signalling messages. The F1-u interface is used to interconnect the O-CU-UP and the O-DU, in order to manage the exchange of user datagrams between the two components. Finally, the E1 interface, interconnects the O-CU-CP and the O-CU-UP, while enabling the exchange of signalling messages. These interfaces have similar protocol stacks as those referred to in Section 2.2. The control plane interfaces have a protocol stack similar to the N2 interface, only the top layer employs a specific application protocol for each interface. The user plane interfaces have a protocol stack similar to the N3 interface, relying on the GTP-U tunnels to transmit user data.

Besides the 3GPP interfaces, the O-RAN architecture includes other interfaces that are standardised by the O-RAN Alliance. Herein, we cover the main ones, such as the A1 interface. The A1 interface connects the Non-RT-RIC to the Near-RT RIC and it supports three types of services: 1) Policy Management Service, in order to optimize the Near-RT RIC and the RAN performance according to the targeted requirements defined in the Non-RT-RIC; 2) Enrichment Information Service, which provides the Near-RT RIC with additional information, in addition to the information that is generally available to it, in order to enhance the performance of its tasks; and 3) ML Model Management Service, allowing ML models to be transferred between the two types of RICs, trained on the Non-RT-RIC and executed on the Near-RT RIC [11]. The protocol stack of the A1 interface is depicted in Figure 2.16.

The E2 interface connects the Near-RT RIC to the E2 nodes (O-CU-CP, O-CU-UP, and O-DU), and the E2 functions are grouped into two categories. The first category is related to the Near-RT RIC Services (REPORT, INSERT, CONTROL, and POLICY), which allow for the exposure of the selected E2 Node data, such as network measurements as well as configuration and control

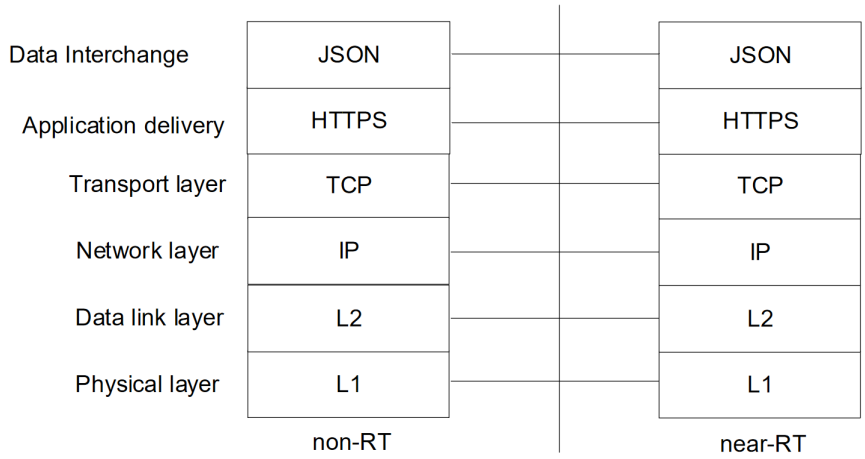

Figure 2.16: A1 protocol stack [11].

actions. The second category includes Near-RT RIC support functions, which mainly include E2 interface management actions (e.g., E2 Setup and E2 Reset) [12]. The protocol stack of the E2 interface is depicted in Figure 2.17.

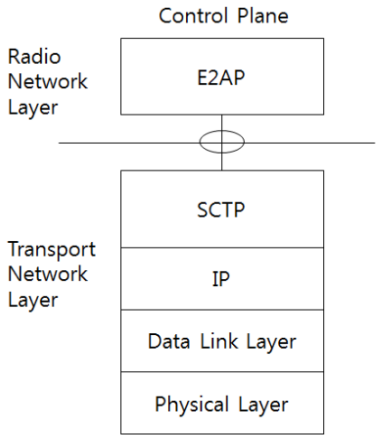

Figure 2.17: E2 protocol stack [12].

In order to connect the O-DU to the O-RU, the O-RAN Alliance defines the Open Fronthaul (FH) Interface. This interface contains the Management (M) Plane and the Control User Synchronization (CUS) Plane. The Open FH M-Plane manages multiple operations related to the life cycle of the RU, such as start-up installation. It also manages the network device's configuration, such as software management, performance management, and configuration management. To perform these tasks, it relies on the Network Configuration Protocol (NETCONF) running on top of an encrypted connection, using Secure Shell Protocol (SSH) or Transport Layer Security (TLS) [29]. The FH CUS-Plane can be decomposed in the Control (C) Plane, the User (U) Plane and the Synchronization (S) Plane. The C-Plane is responsible for transferring commands from the high-PHY in the O-DU to the low-PHY in the O-RU, such as commands for scheduling and beamforming.

The C-Plane relies on whether the Enhanced Common Public Radio Interface (eCPRI) or the Radio over Ethernet (RoE) protocols, in order to ensure communications, which may be used over Ethernet L2 in the case of the O-RAN architecture. Generically, eCPRI can be used over User Datagram Protocol (UDP), but when it comes to O-RAN, UDP is not recommended due to the strict bandwidth and latency requirements [13]. The protocol stack of the C-Plane interface is depicted in Figure 2.18. The U-Plane is in charge of exchanging user data in the frequency domain between the O-DU and the O-RU, using In-phase and Quadrature (I/Q) components, using split option 7.2x. It uses the same protocols as the C-Plane only the payload is different and its protocol stack is similar to the C-Plane stack shown in Figure 2.18.

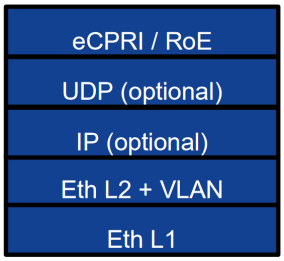

Figure 2.18: C-Plane and U-Plane protocol stack [13].

The last component of the Open FH Interface is the S-Plane. It uses Synchronous Ethernet (SyncE) and Precision Time Protocol (PTP) to synchronize the O-DU and O-RU clocks. Using a master-slave architecture, the master clock, typically on the O-DU, can be used to synchronize the slave clocks, on the O-RU. Despite PTP being able to synchronize clock time/phase and frequency, SyncE is recommended by ITU Telecommunication Standardization Sector (ITU-T) to synchronize clock frequency, in order to obtain better synchronization. The protocol stack of the S-Plane interface is depicted in Figure 2.19.

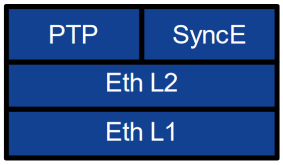

Figure 2.19: S-Plane stack [13].

2.5 Related Work

Since O-RAN standards are recent, there is a relatively low number of related works in the literature regarding O-RAN implementations.

ColO-RAN [30] is a publicly-available wireless network emulator based on dozens of Software-Defined Radios (SDRs). It is implemented over the Colosseum testbed, by means of instantiating a Near-RT RIC based on the O-RAN Software Community (OSC) RIC. The base stations leverage a joint implementation of the 3GPP DUs and CUs, which are implemented using SCOPE [31], an

extended version of srsRAN. srsRAN is an open-source 5G and 4G RAN software package, which allows users to implement gNBs, eNBs, and UEs.

In [32], the authors have implemented an open-source Open RAN use case in the POWDER mobile and wireless research platform, focused on network slicing. They have performed their tests in POWDER's indoor over-the-air lab and POWDER's controlled RF environment. Similarly to ColO-RAN, the authors have implemented the OSC RIC. The base stations are implemented using an extended version of srsRAN, which includes an E2 agent in the srsRAN base station.

In [33], the authors have implemented their own O-RAN based testbed, with one base station and three UEs, while defining different slices for each UE. Similarly to the works presented in [30, 32], the authors have implemented an OSC RIC and the base station relies on an extended version of srsRAN, that includes the E2 interface.

FlexRIC [34] is a software development kit that enables the implementation of specialized service-oriented controllers. FlexRIC's real-time (RT) controller serves the same purpose as the O-RAN based Near-RT RIC. However, according to its authors, FlexRIC is much less computationally demanding. FlexRIC is connected to the base station through the O-RAN-defined E2 interface and is able to monitor and control the RAN. FlexRIC implementations are able not only to connect to a base station, implemented using srsRAN with an E2 Agent, but also simultaneously connect to another base station implemented by means of the Open Air Interface (OAI) software package. OAI allows the deployment of a 4G and 5G RANs, as well as 4G and 5G Core Networks.

ProSLICE [35] disaggregates the RAN itself, which is an aspect that none of the previously mentioned works has addressed. The srsRAN RAN is originally implemented in a monolithic manner, which means it implements the entire 5G NR protocol stack without the ability to split it into different components. The OAI RAN allows for the separation between the CU and DU, but it only supports a single DU connected to each CU. Still, according to the concept of RAN disaggregation, each CU should be able to be connected to multiple DUs. The ProSLICE platform aims for the full-scale realization of the complete O-RAN specification, by extending and enhancing both the OAI RAN software package and the OSC RIC's capabilities. Regarding the OAI RAN, the authors have implemented the disaggregated CU, splitting the CU provided by OAI into the O-CU-CP and O-CU-UP components, implementing the E1 interface, adding support for multiple DUs, and implementing an E2 Agent onto the E2 nodes (O-CU-CP, O-CU-UP, and O-DU). Besides this, the authors have introduced several extensions and enhancements to the OSC RIC, mainly xApps that allow for network slice creation. Despite the fact that the ProSLICE platform is based on open-source components, the software packages, are not available so far.

The concept of mobile gNBs was explored in [36], where the author implements a 5G SA network using OAI. The proposed solution deploys the gNB connected to the Core network via a wireless link, and on-board a mobile robotic platform, a Go1 Edu Robot. To control the position of the mobile gNB, the author developed an On-Demand Mobility Management Function deployed as a Network Function into the Core network, which enables the operator to monitor the QoS of the radio link between the UE and the gNB, and manually control the position of the mobile robotic platform taking advantage of the video feed from the cameras incorporated in the Go1 Edu

Robot.

Even though there are no complete implementations of O-RAN so far, the research into related work presented herein allowed us to identify the tools available in the literature to deploy an O-RAN based network, namely, the two open-source 5G RAN software packages, srsRAN and OAI. For that purpose, they typically use the Universal Software Radio Peripheral (USRP) B210 and the USRP X310 SDRs.

Regarding the concept of mobile gNBs, the goals of this dissertation mostly align with the ones achieved in [36], with the main differences being the introduction of the O-RAN specifications and architecture, the integration of the mobility management function into the O-RAN architecture, instead of deploying it in the Core network, and the goal to develop the mobility function as an autonomous process, that is able to reposition the mobile robotic platform without user input.

To the best of our knowledge, there are no solutions in the literature that address the specific objective of this dissertation of merging the O-RAN architecture with the concept of mobile gNBs, which allows us to potentially develop new contributions for 5G O-RAN networks.

Chapter 3

System Specification, Design, and Implementation

In this chapter, we present the system specification, design and implementation that was performed in order to develop the proposed solution.

3.1 System Specification

The main goal of this dissertation was to implement a private 5G network with a mobile radio cell able to provide wireless connectivity to a UE. This network should be deployed following an SA architecture. The mobile radio cell should be able to autonomously reconfigure and reposition itself, in order to continuously provide wireless connectivity to the UE while being able to adapt to changes in the environment that can hinder the UE's connection to the network.

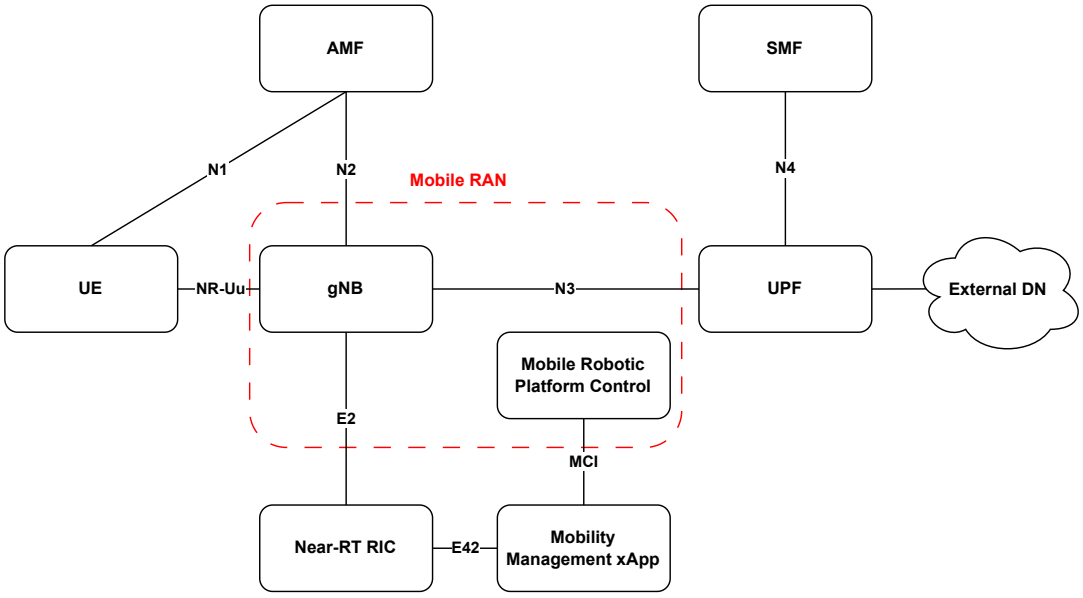

Figure 3.1: System architecture of the proposed solution.

In our proposed architecture (cf. Figure 3.1), a UE needs 5G connectivity to establish a connection to an external Data Network. To ensure this, the UE first establishes a connection with a mobile RAN, which consists of a gNB placed and carried by a mobile robotic platform. The gNB is equipped with radio antennas, which enable transmitting and receiving 5G radio signals.

After the establishment of the connection between UE and gNB, the UE needs to communicate with the AMF, in the 5G Core network, via the N1 interface described in Section 2.2 (cf. Figure 2.6). Using the N1 interface, the UE is able to exchange signalling messages with the AMF, including authentication messages. The connection between UE and AMF is not a direct connection. The N1 interface is the reference for the protocols implemented at each end of the connection, but all messages need to be forwarded by the gNB via the N2 interface.

The N2 interface, also described in Section 2.2 (cf. Figure 2.7), allows for connectivity between the RAN and the Core Network, through the AMF. This connection needs to be established and running before the UE initiates the connection to the 5G network; otherwise, any traffic from the UE to the AMF is not forwarded by the gNB, and the connection between the UE and the 5G Network is not possible.

After successful authentication in the Core Network, the UE must be able to access other external data networks. To achieve this, the mobile RAN needs to communicate with the UPF, which acts as a gateway for the data traffic forwarded to/from the 5G network. This connection is performed via the N3 interface, as explained in Section 2.2 (cf. Figure 2.8).

The UPF also needs to communicate with the SMF, in the 5G Core network, in order to exchange signalling messages and receive control actions associated to the session management of the data flows exchanged between the UE and external networks. This is ensured via the N4 interface, which was described in Section 2.2 (cf. Figure 2.9).

Since in our solution the RAN is mobile, the connection between the RAN and the 5G Core Network is wireless based. This does not pose additional challenges since, as we showed in Section 2.2, both the N2 interface (cf. Figure 2.7) and the N3 interface (cf. Figure 2.8) operate under the TCP/IP protocol stack, which means that any wireless communications technology employing the TCP/IP protocol stack can be used.

Since in our solution the mobile RAN needs to be repositioned on-demand according to the network conditions, there is a need to deploy a Mobility Management Function that can control the position of the mobile robotic platform. For that purpose, we can take advantage of the Near-RT RIC, specified in the O-RAN architecture, to deploy our mobility management function as an xApp. The xApps communicate with the Near-RT RIC via the E42 interface, a non-standardized interface similar to the E2 interface. Since one of the main goals of the Near-RT RIC is to collect information about the status of the network, we can take advantage of this functionality, in order to feed our mobility management function with metrics regarding the UE connection status. These metrics allow us to compute a suitable placement for the mobile RAN.

In order to ensure a connection between the gNB and the Near-RT RIC, we implemented an E2 interface between the two nodes, which was explored in Section 2.4 (cf. Figure 2.17). This allows us to receive metrics regarding the status of the network in the Near-RT RIC, which can

be considered by our xApp. In our solution, the E2 interface leverages on wireless connectivity. For that purpose, any wireless communications technology using the TCP/IP protocol stack can be used (cf. Figure 2.17).

To control the placement of the mobile RAN, our xApp should be able to communicate with the mobile robotic platform. To that end, we propose a novel Mobility Control Interface (MCI), which connects the xApp in charge of implementing the mobility management function to the function responsible for controlling the movement of the mobile robotic platform (cf. Figure 3.1). This interface is characterized by a reliable connection, ensured via TCP, able to exchange data between the two nodes (cf. Figure 3.2).

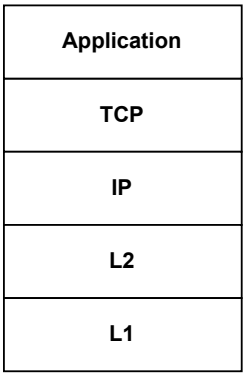

Figure 3.2: Proposed MCI protocol stack.

Since the mobility control of the mobile robotic platform is ensured by an xApp that interacts with a Near-RT RIC, an alternative approach would be to implement the interface between the mobile robotic platform and the mobility management xApp via the E2 interface. However, since most of the commercial mobile robotic platforms available already possess well-defined APIs or Software Development Kits (SDK), which allow for autonomous control via software applications, for simplicity, we took advantage of them.

3.2 Proposed Mobility Management xApp

Our Mobility Management xApp collects metrics from the RAN via the Near-RT RIC and implements a new algorithm that is able to make mobility decisions based on those metrics. This allows our solution to autonomously place the mobile RAN in a suitable position to provide better 5G connectivity to a UE, by increasing the connection SNR. Herein, we explain the new algorithm implemented in the proposed Mobility Management xApp.

Among the metrics that are available to be extracted from the Mobile RAN via the Near-RT RIC software, the one with the most relevance to assess the quality of the connection established between the gNB and the UE is the Signal-to-Noise Ratio (SNR). In addition to SNR, we also retrieve the number of transmitted MAC SDUs for both Uplink (UP) and Downlink (DL) traffic, in order to know if there is a relevant amount of data being exchanged in the network at each moment.

In order to clarify some design choices of the developed algorithm, it is worth considering that:

1. The Mobility Management xApp is not aware of the UE's location.
2. The Mobility Management xApp is only able to collect Near-RT SNR values associated with the current connection between the gNB and the UE.
3. The Mobility Management xApp is unable to assess if changes in SNR are caused due to distance variations or due to obstructions between the Mobile RAN and the UE.

The pseudocode for the implemented algorithm is presented in Algorithm 1. It assumes the mobile RAN has only two possible positions: 1) a default position, which is the position of the gNB when the 5G Network is launched; and 2) a second position, which is the gNB's target position defined by the Mobility Management xApp.

The notation used in Algorithm 1 is detailed in Table 3.1:

Notation	Definition
$\overline{SNR}$	Average SNR measured in the current position in the last N seconds (e.g., 5 seconds) .
A_SNR	Minimum SNR value to consider the channel quality as acceptable
$P\_\overline{SNR}$	$\overline{SNR}$ value measured in the previous position before the change to the current position.
$Threshold$	Value of the difference between $P\_\overline{SNR}$ and $\overline{SNR}$ bellow which it is not worth changing the positioning.
$t\_Position$	Timestamp indicating when the change to the current position happened.
t_Now	Timestamp indicating the current time.
T_1	Maximum value in seconds during which the Mobile Robotic Platform should stay in the current position if the $\overline{SNR}$ value is lower than A_SNR.
T_2	Maximum value in seconds during which the Mobile Robotic Platform should stay in the second position, even if the $\overline{SNR}$ value is higher than A_SNR.
N_SDUs	Number of transmitted DL and UP MAC SDUs in the last N seconds.
$SDU\_Threshold$	Maximum number of MAC SDUs transmitted, in order to infer the volume of traffic exchanged with the UE.

Table 3.1: Defenition of the variables used in algorithm Algorithm 1.

The algorithm first checks in line 1 of Algorithm 1, if the value $\overline{SNR}$ is lower than the minimum acceptable value A_SNR, as well as lower than the $P\_\overline{SNR}$ minus a specific $Threshold$ value. The latter check allows for avoiding moving the mobile robotic platform if the channel quality in the

current position is better than the last recorded channel quality in the previous position. The $Threshold$ is necessary because if the $\overline{SNR}$ is just slightly worse than in the previous position, it might not be worth it to move the Mobile Robotic Platform. If both conditions are true, then the algorithm updates variables and moves the Mobile Robotic Platform to the other position.

In case the $\overline{SNR}$ is less than A_SNR but still higher than $P\_\overline{SNR}$, the Mobile Robotic Platform stays in the same position until T_1 seconds pass, as shown in line 5. After this time, the mobile robotic platform moves to the other position because, in the period of time that passed, the channel quality in the other position might have improved and actually be acceptable.

Lastly, it might be advantageous for the Mobile Robotic Platform to be in the default position (e.g., for charging). As such, in line 9, the algorithm checks if the Mobile Robotic Platform has been in the second position for longer than T_2 and if the network is not being heavily used, by assessing if the N_SDUs is lower than $SDU\_Threshold$. Then, the Mobile Robotic Platform moves to the default position, even if the channel quality is acceptable in the current position.

If none of the previous conditions are met, the Mobile Robotic Platform stays in its current position.

Algorithm 1 Algorithm implemented in the Mobility Management xApp for two positions.

1: **if** $\overline{SNR} < A\_SNR$ **AND** $\overline{SNR} < P\_\overline{SNR} - Threshold$ **then**
2: $P\_\overline{SNR} \leftarrow \overline{SNR}$
3: $t\_Position \leftarrow t\_Now$
4: Move to the other position
5: **else if** $\overline{SNR} < A\_SNR$ **AND** $t\_Now - t\_Position > T\_1$ **then**
6: $P\_\overline{SNR} \leftarrow \overline{SNR}$
7: $t\_Position \leftarrow t\_Now$
8: Move to the other position
9: **else if** $t\_Now - t\_Position > T\_2$ **AND** $N\_SDUs < SDU\_Threshold$ **AND** Current_Position == Second_Position **then**
10: $P\_\overline{SNR} \leftarrow \overline{SNR}$
11: $t\_Position \leftarrow t\_Now$
12: Move to default position
13: **else**
14: Stay in Position
15: **end if**

3.3 System Design

The architecture of the system that was designed to implement and evaluate the proposed solution is depicted in Figure 3.3. The system is composed of three main logical units. One unit in which we implement the 5G Core Network, the Near-RT RIC and the Mobility Management xApp. Another implements the mobile RAN, which consists of the gNB, the USRP SDR board, the Robotic Control application, and the Mobile Robotic Unit. Moreover, the system includes a unit, which implements the UE software, connected to a USRP SDR board.

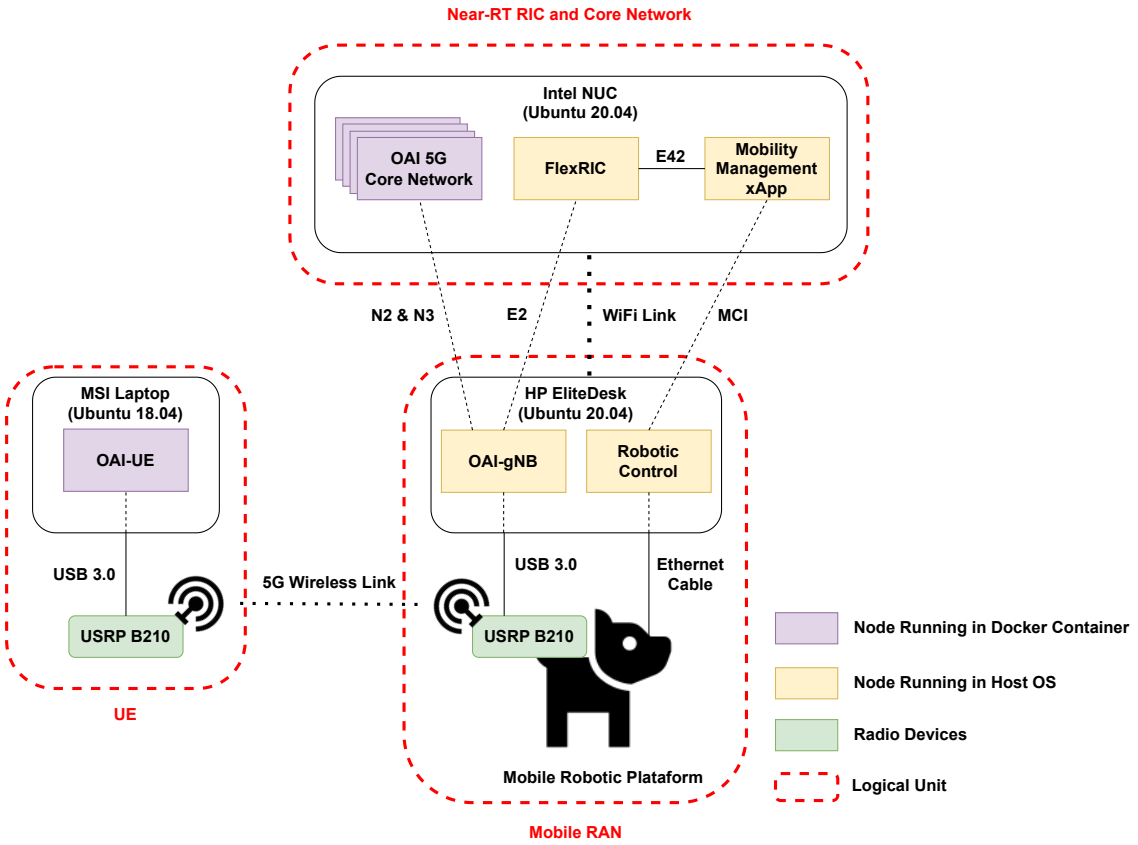

Figure 3.3: Architecture of the system designed to implement and evaluate the proposed solution.

In the following sections, we will detail the hardware used to implement each component of the system, as well as explain the choices of software packages that were used.

3.3.1 Software Packages

3.3.1.1 5G Core Network, 5G gNB, and 5G UE

The main open-source 5G software packages available to implement an O-RAN based architecture are OAI [37] and srsRAN [38]. In order to implement our system, the software package chosen was OAI for three reasons. First, OAI provides all the components required to deploy a 5G SA network, including the RAN and Core networks; srsRAN software only allows the deployment of the RAN, introducing the need to implement the Core network using a different software package, such as OAI. Second, we wanted to explore the concept of RAN splitting. At the time of designing our system, srsRAN did not allow splitting the gNB into CU and DU, as OAI does. Despite this, both OAI and srsRAN developments were evolving towards an O-RAN based architecture [39] [40]. Third, in order to connect the Near-RT RIC with either OAI or srsRAN, we needed to install a patch provided by the RIC's developers onto the RAN software. This patch was crucial for the proper installation and configuration of an E2 node within the RAN. Without the patch, establishing a connection between the Near-RT RIC and the RAN was not possible. However, the

patch for srsRAN is currently available only for the software that implements an eNB; it is not compatible with the software that implements the gNB. On the other hand, the patch provided by OAI works for both eNB and gNB. This distinction implies that if we selected srsRAN, we would be limited to a 4G network, whereas OAI allows us to implement a 5G network.

3.3.1.2 Near-RT RIC

Two main open-source software packages are able to implement a Near-RT RIC: O-RAN Software Community's Near-RT RIC [41] and Mosaic5G's FlexRIC [42]. Considering these two software packages, the one chosen to implement the Near-RT RIC was Mosaic5G's FlexRIC. There are several reasons for this choice. First, the system requirements in order to run OSC's Near-RT RIC are relatively high [43], while for FlexRIC there are no specific system requirements listed; only a few software prerequisites are mentioned. FlexRIC is a lightweight application that can be launched from an executable file, which also makes its installation easier. Second, FlexRIC's documentation states that it allows the users to extract measurements from different layers of the 5G protocol stack of the RAN, such as MAC, RLC and PDCP layers. In OSC's Near-RT RIC documentation, there is no explicit mention of specific capabilities apart from some test scenarios. Besides this, FlexRIC provides a new interface named E42, which is used to exchange data between the xApps and the FlexRIC. The mode of operation is similar to the E2 interface. For example, if an xApp needs to send a request for metrics to an E2 node, like a gNB, it can send that request through the E42 interface to FlexRIC. Then, FlexRIC forwards the request via the E2 interface to the gNB, acting as an intermediary between the xApp and the gNB. This interaction is depicted in Figure 3.4.

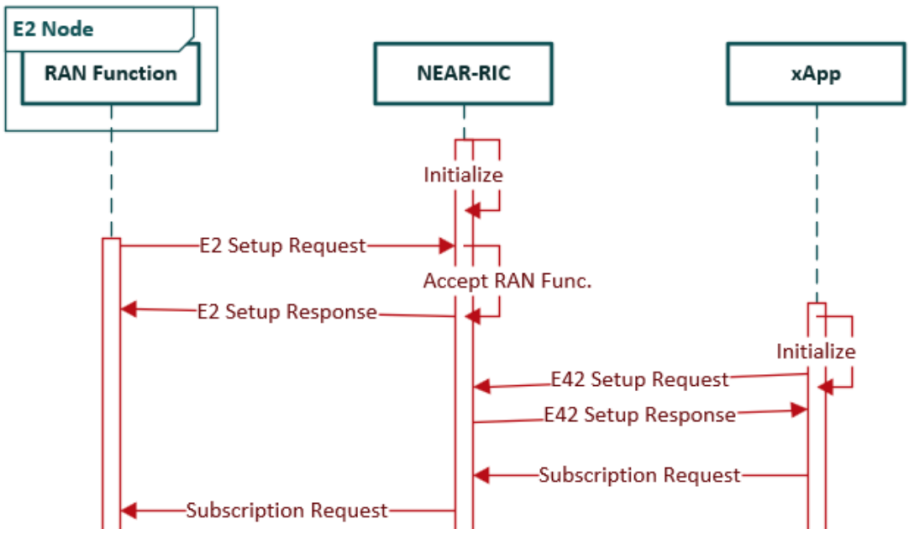

Figure 3.4: E42 interface logical operation, adapted from [14].

The E42 interface can be implemented using a simple API that allows users to develop applications in C and other programming languages; these applications can be used to extract metrics from the RAN connected to FlexRIC. This allows users to implement any xApp to perform any

desired task, as long as the required interactions between the xApp and the RAN are supported by the FlexRIC software.

3.3.2 Hardware

3.3.2.1 Core Network and FlexRIC

Both the OAI 5G Core Network and FlexRIC were deployed in an *Intel NUC* (Next Unit of Computing) *Board NUC5i5MYBE* [44] depicted in Figure 3.5.

Figure 3.5: Intel NUC Board NUC5i5MYBE.

The OAI 5G Core Network was deployed using Docker containers. Since the demanded computational requirements are low, it is suitable for deployment in less powerful computers, such as the Intel NUC.

Similarly, the FlexRIC is not resource intensive, as explained in Section 3.3.1.2. For this reason, it can be executed alongside the OAI Core Network on the Intel NUC.

The Mobility Management xApp is also deployed alongside FlexRIC, as it needs to be running in the same computing unit in order to properly use the E42 interface and to reduce latency for exchanging data with the xApp.

As explained in Section 3.1, both the connection between the RAN and the Core Network, using the N2 and N3 interfaces, and the connection between the RAN and the Near-RT RIC, through the E2 interface, use the TCP/IP protocol stack. Considering this, in order to ensure wireless connectivity between the Core Network and the Mobile RAN, a wireless link was established. The Wi-Fi technology was chosen due to its wide support and ease of implementation.

3.3.2.2 gNB and UE

The gNB was deployed in an *HP EliteDesk 800 G2 Small Form Factor* (SFF), with 4 physical CPU cores and 16 GB of RAM. While not ideal to run the gNB according to the recommended system requirements of 8 physical CPU cores and 32 GB of RAM, this processing unit proved to be enough to run the OAI gNB software. As we can see in Figure 3.6, even though the HP EliteDesk is a Small Form Factor and not a Full Tower Desktop, it is still much larger than the Intel NUC; for this reason, it is not suitable to be placed on-board the Mobile Robotic Platform. However, since the mobile RAN is supposed to be a proof of concept and not finalized commercial unit, this setup does not compromise validating the proposed solution, by placing only the SDR board on the Mobile Robotic Platform, while controlling its position.

Figure 3.6: HP EliteDesk 800 G2E SFF.

Besides the gNB, our Robotic Control application was also deployed in the HP EliteDesk. This is due to two reasons. First, the SDK to control the Mobile Robotic Platform is implemented and intended to be used with C++ applications, while the API provided by FlexRIC is intended to work with xApps implemented in C or Python. Because of this, we implemented the Mobility Management xApp in C; it runs alongside the Core Network and FlexRIC on the Intel NUC. This setup enables the xApp to send control actions to the Robotic Control application, written in C++, which runs alongside the gNB on the HP EliteDesk. The communications between them use TCP sockets via the proposed MCI interface (cf. Figure 3.2). The Robotic Control application is in charge of controlling the movement of the Mobile Robotic Platform, based on the commands received from the Mobility Management xApp.

The second reason for deploying the Robotic Control application on the HP EliteDesk is related to the connection with the Mobile Robotic Platform itself. Despite the chosen Mobile Robotic Platform offering both a wired connection using Ethernet or a wireless connection through Wi-Fi, we established an Ethernet-based connection. This approach was preferred over a Wi-Fi connection to the Mobile Robotic Platform to avoid managing two separate Wi-Fi links on the HP EliteDesk, which would significantly increase complexity.

The OAI 5G UE software was deployed in an *MSI GE75 Raider* Laptop, depicted in Figure 3.7, with 6 physical CPU cores. The UE is computationally demanding, with recommended requirements of 8 physical CPU cores. Additionally, the UE application is more sensitive to communication errors or delays between the computer hosting it and the SDR, which occur relatively frequently. In contrast, the gNB application can recover from such errors and delays most of the time. It is worth noting that the UE application tends to crash or lose connection to the gNB, requiring frequent relaunching. By deploying the UE application on the computer with more resources, we can reduce the risk of errors or delays occurring between the computer and SDR. This minimizes the need to relaunch the UE application, making testing more convenient and less time-consuming.

Figure 3.7: MSI GE 75 Raider Laptop.

Since the RAN and the UE were deployed in separate host computers, two SDRs were used to enable a 5G radio connection between the gNB and the UE. As we concluded from the related implementations of other 5G networks in Section 2.5, the most commonly used SDRs are the USRP B210 [45] and the USRP X310 [46]. According to the OAI wiki [47], these two SDR models are recommended for implementing OAI based networks. For our system implementation, we selected the USRP B210 SDR, due to its popularity across the community and its cost-effectiveness. This particular model, depicted in Figure 3.8, utilizes a Universal Serial Bus (USB) 3.0 interface to establish a connection with the computer acting as a processing unit.

Figure 3.8: USRP B210 with the respective USB 3.0 interface.

The carrier frequency selected to implement the 5G RAN was 3.6 GHz. We used two W5084K [48] dipole antennas tailored to this frequency band, which allowed for an optimized wireless connection between the two SDRs acting as gNB and UE, respectively. The two SDRs with the respective antennas attached are depicted in Figure 3.9.

Figure 3.9: USRP B210 SDR devices with the respective W5084K antennas.

3.3.2.3 Mobile Robotic Plataform

The chosen platform to serve as the Mobile Robotic Platform acting as the Mobile RAN was the Unitree's GO 1 Robot [15] depicted in Figure 3.10.

Figure 3.10: Unitree's GO 1 robot [15].

Besides being a capable robotic unit, able to carry payload, the main reason for choosing the GO 1 robot is the fact that it can be controlled autonomously via simple user-made applications, which is fundamental in order to implement the proposed solution. In order to support this, Unitree provides an SDK, the *unitree_legged_sdk*, which allows for both high-level and low-level control of the GO 1 robot. This SDK allows users to implement C++ programs that are able to communicate and control the GO 1 robot. Our Robotic Control application is a C++ application that receives mobility control commands from the Mobility Management xApp, via the MCI interface (cf. Figure 3.2) using TCP sockets, and translates those commands into actual movement instructions for the GO 1 robot using the *unitree_legged_sdk*.

The GO 1 robot also hosts a Web page provided by a Hypertext Transfer Protocol (HTTP) server running in the GO 1 robot. The web page can be accessed in a browser on any computer connected to the GO 1 robot, and it allows the user to see the video feed in real-time from the cameras installed on the GO 1 robot, as well as control the movement of the robot using the WASD keyboard keys to move the mobile robotic platform front, left, backwards and right. This service may be useful if a user needs to manually fine-tune the position of the Mobile Robotic Platform.

3.4 System Implementation

In this section, we present the main aspects related to the implementation of our testbed, especially changes that were performed in the configuration files, in order to properly run each node, changes

performed on the Host OS, as well as general setup requirements.

3.4.1 OAI 5G Core Network

In order to implement the 5G Core Network, OAI provides a few different implementation methods, including Kubernetes cluster and Docker containers. To deploy our Core Network we chose to implement it using Docker containers, due to the fact that this option is the one with the most support and tools available from the OAI software package and due to its ease of implementation.

OAI offers different Core Network options that allow the implementation of 5G Core Network Functions based on the user's needs. To deploy our testbed we opted for the minimal deployment, due to the fact that it included every Network Function we needed to implement our proposed solution and allowed us to reduce computational resources needed, as it uses fewer containers, allowing for increased performance on the Core Network. The minimal deployment includes 5 Docker containers, the AMF, SMF, UPF, Network Repository Function (NRF), and MySQL.

The functions of the AMF, SMF, and UPF were already covered in Section 2.2. The NRF is also a Network Function included in 5G reference architecture (cf. Figure 2.4); as the name suggests, NRF is responsible for holding information concerning other Network Functions present in the 5G Core Network. When a new Network Function is deployed in the Core Network, it registers itself with the NRF, and obtains information about other Network Functions present in the Core Network from the NRF, such as their IP addresses. With this information, the new Network Function can then establish direct connections with the remaining Network Functions. The MySQL Docker container is mainly used to store information pertaining to the identification of the UEs that are registered in the Core Network.

Besides the 5 containers mentioned, OAI also provides an extra container, named *ext-dn*, which acts as a computer located on an external DN connected to the UPF. This container is useful in order to conduct performance tests between UEs and an external DN, without the influence of external networks impacting the results of our testing.

After pulling V1.5.0 of the docker images and downloading the remaining files needed, such as the *.sql* files for the database and the *docker-compose.yaml* file, which is used to configure and instantiate the Core Network, we configured the Host OS in order to properly forward traffic to and from the Core Network containers. To achieve this we changed IP forwarding rules for the Linux Host, and the Iptables policies towards docker containers, in order to accept packets from external connections, as depicted in Figure 3.11.

```
Core-Network-Host:$ sudo sysctl net.ipv4.conf.all.forwarding=1
net.ipv4.conf.all.forwarding = 1
Core-Network-Host:$ sudo iptables –P FORWARD ACCEPT
```

Figure 3.11: Linux IP forwarding and Iptables configurations.

In order to configure the Core Network, the *docker-compose.yaml* file was used. This file contains parameters that are passed to each Docker container when they are initialized. The main parameters that require the user's attention are the Public Land Mobile Network (PLMN) ID which consists of two parts: 1) the Mobile Country Code (MCC) and the Mobile Network Code (MNC), which are mainly used to identify a specific network; and 2) the Tracking Area Code (TAC), which is used to identify the area a gNB belongs to – the Core Network can manage multiple gNBs in different locations, which are grouped in different areas identified by their TAC. The actual values chosen for these fields are not important unless they are reserved values, since we are dealing with a private 5G Network. However, it is important to ensure that the values configured in the Core Network match the values configured in the gNB; otherwise, the connection between the two might not be successful, especially if the TAC values do not match. The values used in the *docker-compose.yaml* file to deploy our testbed were 208 for the MCC, 95 for the MNC, and 0xA000 for the TAC, as depicted in Figure 3.12.

```
version: '3.8'
services:
(...)
    oai-amf:
        container_name: "oai-amf"
        image: oaisoftwarealliance/oai-amf:v1.5.0
        environment:
            (...)
            - MCC=208
            - MNC=95
            (...)
            - PLMN_SUPPORT_MCC=208
            - PLMN_SUPPORT_MNC=95
            - PLMN_SUPPORT_TAC=0xa000
```

Figure 3.12: Defined MNC, MCC, and TAC values for AMF container in the *docker-compose.yaml* file.

When a UE connects to the 5G Network, it gets an IP address that is used to send and receive the User Plane traffic. Other external Data Networks do not know the structure of the Private 5G network. As such, we did not have defined routes in order to reach the UEs based on their IP inside the 5G Network. To that end, the UPF, acting as a gateway between the 5G Network and other external Data Networks, must perform Network Address Translation (NAT) on the UE IPs, in order to enable communication with external networks. This configuration must also be set in the *docker-compose.yaml* file. Figure 3.13 depicts the configuration needed for NAT as well as the UE IP pool used in our 5G network.

```
version: '3.8'
services:
(...)
    oai-spgwu:
        container_name: "oai-spgwu"
        image: oaisoftwarealliance/oai-spgwu-tiny:v1.5.0
        environment:
            (...)
            - NETWORK_UE_NAT_OPTION=yes
            - NETWORK_UE_IP=12.1.1.0/24
```

Figure 3.13: UE IP pool and configuration UPF the perform NAT in the *docker-compose.yaml* file.

After properly configuring the *docker-compose.yaml* file with the mentioned fields, required docker networks, container IP addresses, and routes, we can launch our 5G Core Network. When it is launched, besides creating the containers, it also creates two interfaces on the Host OS, in order to interact with the two Docker Networks. In our case, they are named *demo-oai* for the network hosting the containers belonging to the Core Network and *oai-traffic*, which connects the UPF to the *ext_dn* container, as depicted in Figure 3.14.

```
Core-Network-Host:$ ifconfig
demo-oai: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.70.129  netmask 255.255.255.192  broadcast 192.168.70.191
        inet6 fe80::42:29ff:fef3:79cb  prefixlen 64  scopeid 0x20<link>
(...)
oai-traffic: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.71.129  netmask 255.255.255.192  broadcast 192.168.71.191
        inet6 fe80::42:d3ff:fe1f:cbf2  prefixlen 64  scopeid 0x20<link>
```

Figure 3.14: Interfaces created in Host OS to interact with Docker Networks.

If everything was properly configured, and the docker containers are in a *healthy* state, our OAI 5G Core Network is up and running. A network diagram representing the implemented Core Network is depicted in Figure 3.15. Also, an IP address scheme of the implemented components in the Core Network and Data Network is presented in Table 3.2 and Table 3.3. This information is useful when analysing Wireshark logs, in order to easily identify which Core Network component a packet originated from and which component is its destination.

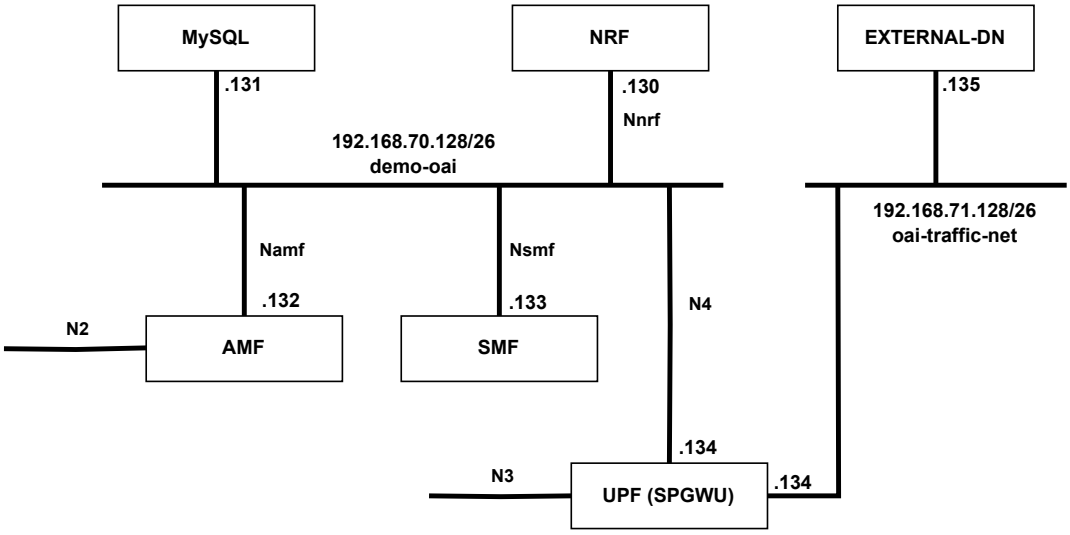

Figure 3.15: Deployed 5G Core Network Diagram.

Core Network Function	IP
NRF	192.168.70.130
MySQL	192.168.70.131
AMF	192.168.70.132
SMF	192.168.70.133
UPF	192.168.70.134

Table 3.2: Deployed 5G Core Network Functions IP addresses.

Data Network Component	IP
UPF	192.168.71.134
EXTERNAL-DN	192.168.71.135

Table 3.3: Deployed Data Network Component IP addresses.

The last configuration needed to fully set up our 5G Core Network is to register the UE on the Core Network. The registration is performed by adding the OAI UE Subscriber Identity Module(SIM) details to the MySQL database, including the International Mobile Subscriber Identity (IMSI), its related key, and the Operator Code (OPc). This can be done by adding the information in the *oai_db.sql* files before starting the Core Network; it can also be performed in runtime, after starting the Core Network, by logging inside the MySQL Docker container as shown in Figure 3.16. The values registered in the Core Network must match the ones used by the UE; otherwise, the UE is not able to authenticate itself in the Core Network.

```
Core-Network-Host:$ docker exec -it mysql /bin/bash
bash-4.4# mysql -uroot -plinux -D oai_db
(...)
mysql> INSERT INTO users VALUES
    -> ('2089500007487','380561234567','55000000000001',NULL,'PURGED',50
    ,40000000,100000000,47,0000000000,1,0xfec86ba6eb707ed08905757b1bb44b8f
    ,0,0,0x40,'ebd07771ace8677a',0xC42449363BBAD02B66D16BC975D77CC1);
Query OK, 1 row affected (0.00 sec)
```

Figure 3.16: Registering UE in Core Network, by adding IMSI, associated Key, and OPc, highlighted in red, respectively, to the MySQL Database.

3.4.2 OAI gNB

In order to deploy the OAI gNB we used a specific version from the OAI Repository [49], more specifically version 2022.41. This version is necessary in order to install the FlexRIC patch that implements an E2 node on OAI's gNB software, as explained in [42].

Even though the computers running the gNB and the Core Network are connected via a Wi-Fi link, since the Core Network is implemented using Docker containers with IP addresses belonging to digital Docker networks hosted on the Intel NUC, the computer hosting the gNB, and consequently the gNB, are not able to reach the components of the Core Network by default, even if given their IP addresses. In order to address the issue, we need to configure routes in the gNB Host, so that it can reach the Core Network, via the Core Network Host, as shown in Figure 3.17.

```
gNB-Host:$ sudo ip route add 192.168.70.128/26 via 10.42.0.16 dev
    wlxc04a001bcb9b
gNB-Host:$ ip r
(...)
192.168.70.128/26 via 10.42.0.16 dev wlxc04a001bcb9b
```

Figure 3.17: Adding route to Core Network in the gNB Host computer, via the Core Network Host computer.

As mentioned in Section 3.4.1, the MCC, MNC, and TAC values must be the same in the configuration files of both the Core Network and the gNB. To this end, we modified the *gnb.sa.band78.fr1.106PRB.usrpb210.conf* configuration file, used to launch the gNB with the particular USRP B210 SDR model we used in our testbed. In Figure 3.18, the MCC, MNC, and TAC values match the ones introduced in the Core Network (cf. Figure 3.12).

```
(...)
gNBs =
(
 {
    ////////// Identification parameters:
(...)
    // Tracking area code, 0x0000 and 0xfffe are reserved values
    tracking_area_code  =  0xa000;
    plmn_list = ({mcc = 208; mnc = 95; mnc_length = 2; snssaiList =({sst=1})});
```

Figure 3.18: Defined MNC, MCC, and TAC values for the gNB in the configurations file.

If the values in the Core Network configuration file and the gNB configuration file do not match, the NGAP setup between AMF and gNB over the N2 interface fails, as shown in the Wireshark capture in Figure 3.19.

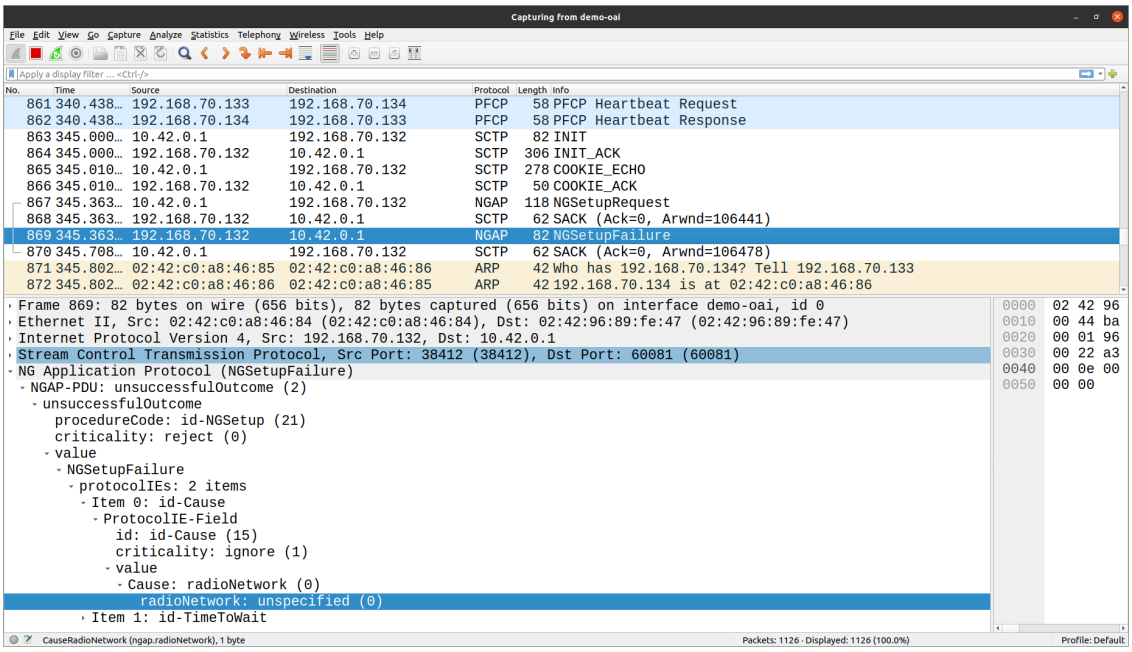

Figure 3.19: NGAP setup failure between gNB and AMF, due to non-matching TAC value.

Besides configuring the proper values for MCC, MNC, and TAC fields, we also need to configure the gNB configuration file with the IP address of the AMF, as well as set the IP that the gNB should use in order to communicate with the AMF and SMF. Since the gNB is running on the Host OS, and the computers that run the Core Network and gNB are connected through a Wi-Fi link, the gNB IP and interface as the host's physical interfaces, as depicted in Figure 3.20.

To enable a connection between the OAI gNB software and the USRP B210 SDR, we utilize the USRP Hardware Driver (UHD), a user-space library that is able to communicate and controls all the devices of the USRP family, allowing users to transmit and receive waveform samples to and from USRP devices, as well as fine-tune radio parameters such as sampling rate, central

```
(...)
    ////////// AMF parameters:
    amf_ip_address      = ( { ipv4       = "192.168.70.132";
(...)
    NETWORK_INTERFACES :
    {
        GNB_INTERFACE_NAME_FOR_NG_AMF            = "wlxc04a001bcb9b";
        GNB_IPV4_ADDRESS_FOR_NG_AMF              = "10.42.0.1/24";
        GNB_INTERFACE_NAME_FOR_NGU               = "wlxc04a001bcb9b";
        GNB_IPV4_ADDRESS_FOR_NGU                 = "10.42.0.1/24";
```

Figure 3.20: Defined AMF IP address and gNB IP address and interface.

frequencies, and gain [50]. This driver also allows the Host OS to detect and load the proper firmware and configuration on the USRP board according to its model. As recommended by OAI, we installed UHD version 4.4.0 and downloaded the proper firmware images for our SDR board. Figure 3.21 shows the UHD driver detecting and the USRP B210 board used and loading the proper firmware.

```
gNB-Host:$ uhd_config_info --version
UHD 4.4.0.HEAD-0-g5fac246b
gNB-Host:$uhd_find_devices
[INFO] [UHD] linux; GNU C++ version 9.4.0; Boost_107100; UHD_4.4.0.HEAD-0-
    g5fac246b
[INFO] [B200] Loading firmware image: /usr/local/share/uhd/images/usrp_b200_fw
    .hex...
- - - - - - - - - - - - - - - - - - - - - -
- - UHD Device 0
- - - - - - - - - - - - - - - - - - - - - -
Device Address:
    serial: 307B5FD
    name:
    product: B210
    type: b200
```

Figure 3.21: UHD discovering the connected USRP B210 board and loading respective firmware.

3.4.3 OAI 5G UE

Similar to the OAI gNB software, the OAI UE software is intended to be launched from an application running in the Host OS. However, even after trying multiple versions and branches of the OAI UE software on three different versions of the Ubuntu OS (18.04, 20.04, and 22.04), which were recommended on different tutorial implementations on the OAi Wiki, we were unable to get the OAI UE fully operational using the intended application.

The OAI UE software uses Linux TUN interfaces, which are used in order to create an IP interface with user space programs [51]; they allow for IP communications between the Host OS running the UE software and the 5G Network. However, in our implementation, despite the UE successfully registering itself in the Core Network, the TUN interface was unable to be created by the software, preventing sending and receiving IP traffic.

In order to overcome this, we used a Docker container to deploy the UE. Besides the Docker containers offered by OAI to implement the Core Network, OAI also has available two additional Docker containers to implement the gNB and UE. These two containers are only mentioned and used when deploying the 5G Network in *rfsimulator* mode. *Rfsimulator* allows emulating a radio interface, by carrying the time-domain samples between UE and gNB over a network interface, without the need for SDRs, which is useful to test the 5G network using just one computer and without the influence of the wireless channel medium. Moreover, if we analyze the *entrypoint.sh* script for the UE Docker container, we can conclude that the software was compiled in order to also allow for deployment with radio units such as the USRP boards. In order to enable the option for the OAI UE Docker container to use the USRP boards, we need to add an environment variable to the *docker-compose.yaml* file that launches the UE Docker container, give the container access to the Host USB ports, in order for the container to communicate with the USRP B210 board, and deploy the container on the Host's network, instead of a Digital Docker network, in order for the Container to create a Linux TUN interface on the Host OS. These necessary changes to the OAI UE *docker-compose.yaml* are depicted in Figure 3.22.

```
version: '3.8'
services:
    oai_nr_ue:
        image: oaisoftwarealliance/oai_nr_ue:develop
        privileged: true
        container_name: rfsim5g_oai_nr_ue
        environment:
(...)
            USE_B2XX: 1
(...)
        devices:
            - '/dev:/dev'
        network_mode: host
```

Figure 3.22: Necessary changes to allow the UE container to access the USRP B210 SDR board.

Besides these changes for using the USRP board, we also need to set the UE SIM values to the same values that were registered in the MySQL database at the Core Network (cf. Figure 3.16); otherwise, the UE is not able to authenticate itself in the Core Network, since the SIM details are not registered. To do this, we also need to add environment variables to the OAI UE's *docker-compose.yaml* file with the IMSI, associated key, and OPc values, as shown in Figure 3.23.

```
version: '3.8'
services:
    oai_nr_ue:
(...)
        environment:
            FULL_IMSI: '2089500007487'
            FULL_KEY: 'fec86ba6eb707ed08905757b1bb44b8f'
            OPC: 'C42449363BBAD02B66D16BC975D77CC1'
```

Figure 3.23: Defining the proper UE SIM details values for the IMSI, associated key, and OPc.

If the OAI UE SIM details do not match the SIM details registered in the Core Network database, the Core Network rejects the UE registration request as depicted in the Wireshark log of Figure 3.24, making it impossible for the UE to connect and send/receive traffic through the 5G Network.

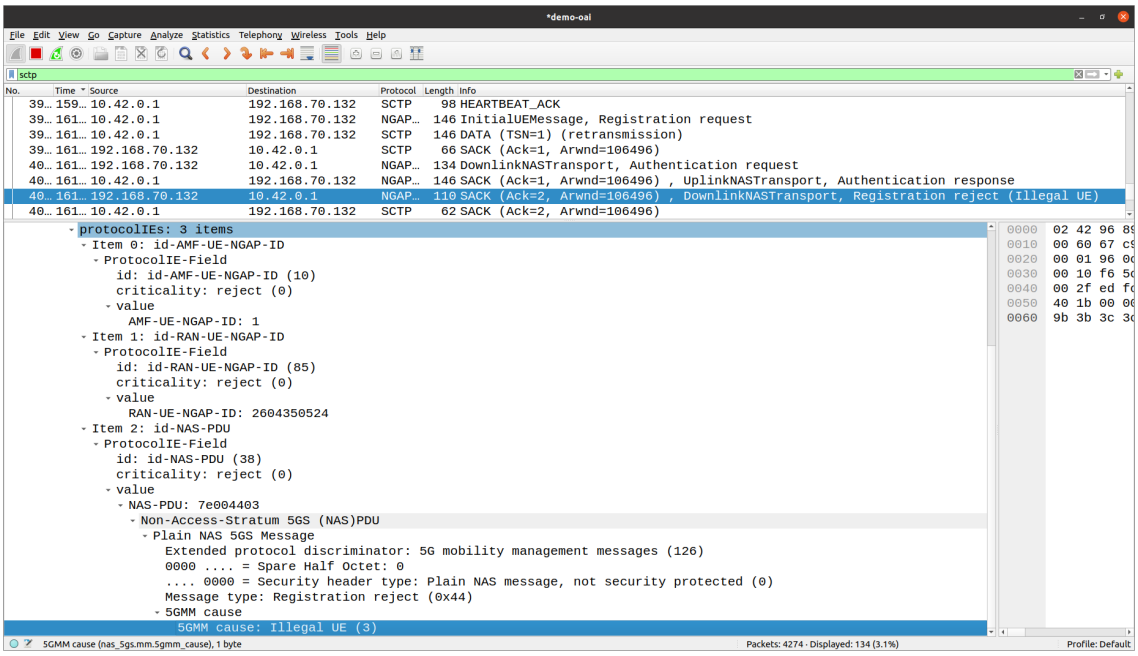

Figure 3.24: Core Network rejecting the UE registration request due to non-matching UE SIM details.

3.4.4 FlexRIC

The FlexRIC application is deployed on the same computer as the OAI 5G Core Network. After applying the FlexRIC patch to the OAI gNB software, as mentioned in Section 3.4.2, installing the required dependencies, and compiling the FlexRIC software, it is only necessary to indicate the IP address to connect to the FlexRIC, both in the FlexRIC Host computer (the Intel NUC) and the gNB Host computer (the HP Elitedesk). This file is located in */usr/local/etc/flexric/flexric.conf* and its default value is the Loopback Interface address (127.0.0.1). The IP address should match the

FlexRIC Host's IP address in the Wi-Fi interface. If this change is not performed in the FlexRIC Host, the FlexRIC application is launched with the IP address of the computer's Loopback Interface address, making it unreachable from outside networks. If this change is not performed in the gNB Host, the gNB will send the E2 Setup Request packets to its own Loopback interface address, which also makes the connection between gNB and FlexRIC impossible. The necessary change to the */usr/local/etc/flexric/flexric.conf* file in both computers is depicted in Figure 3.1.

```
[NEAR_RIC]
NEAR_RIC_IP = 10.42.0.16
#NEAR_RIC_IP = 127.0.0.1 The default IP was defined to the loopback interface
```

Listing 3.1: Configuring FlexRIC's IP in the configuration file of FlexRIC Host and gNB Host.

3.5 Summary

To implement the proposed solution we divided the system into three logical units. The first one consisted of the OAI 5G Core Network, FlexRIC, and Mobility Management xApp, which were deployed on an Intel NUC mini-computer. The second used an HP EliteDesk to host the Robotic Control application and the OAI gNB software connected to a USRP B210 SDR board, placed on top of a Unitree GO 1 robot acting as a Mobile Robotic Platform. The third acted as a UE, by hosting the OAI 5G UE software on an MSI Laptop, which was also connected to a USRP B210 SDR board.

The Intel NUC was connected to the HP EliteDesk via a Wi-Fi link, allowing for the gNB to communicate with the Core Network using the N2 and N3 interfaces, and to communicate with the FlexRIC via the E2 interface. This connection is also used by the Mobility Management xApp, in order to send mobility control commands to the Robotic Control application, through the proposed MCI interface, which in turn converts them into actual movement instructions for the GO 1 robot. The connection between the UE was ensured by a 5G radio connection between the SDRs connected to the UE and gNB. It is important to ensure that every parameter is correctly configured in the Core Network, FlexRIC, gNB, and UE, otherwise one or more aspects of the proposed solution might not work as intended.

It is worth noting that other work, not presented in this Chapter, was conducted but not implemented in the final proposed solution, due to performance or stability issues. In Appendix A, we present some of that work, as well as the reasons for not being included in the final proposed solution.

Chapter 4

System Validation

This chapter focuses on the validation of the proposed solution. Section 4.1 presents the testing methodology. Section 4.2, discusses the performance of the Wi-Fi backhaul link. Sections 4.3 to 4.7 focuses on testing and validating the correct functioning of the individual components of the proposed solution, as well as their correct interoperability. Section 4.8 presents the system validation in a real-world use case scenario. Finally, in Section 4.9, we analyze and discuss the achieved results.

4.1 Metodology

In the system validation phase, a first test was performed, in order to ensure that the Wi-Fi link between gNB and Core Network was not the bottleneck from the network performance point of view. The subsequent tests focused on validating the components of the 5G Network, both individually and jointly, in order to ensure their interoperability. The order in which the tests were conducted is important, as the system deployment must follow a specific order, starting by deploying the Core Network and FlexRIC, followed by the gNB, and finally the UE, Robotic Control application, and Mobility Management xApp. The final test aimed at evaluating the system in a realistic scenario, in order to validate a typical use case for the proposed solution.

The testing methodology consisted of first describing the test itself and the setup used to perform it. This was followed by a functional validation, using tools such as *ping* and Wireshark, in order to check for connectivity and ensure the proper protocols and interfaces were being used. Finally, when applicable, we assessed the network performance by using the *iPerf* tool [52] to measure throughput and packet loss ratio.

4.2 Performance Assessment of Wi-Fi based Backhaul Link

This test was performed in order to ensure that the connection between the UE and an external Data Network was not constrained by the Wi-Fi based backhaul link, used to connect the gNB to

the Core network. This would happen if the throughput achievable on our Wi-Fi based backhaul link was lower than the one achieved in the 5G link between UE and gNB.

In order to ensure that the backhaul link does not lead to a potential bottleneck, we performed two tests. First, we measured, using the *iPerf* tool, the throughput achieved in the Wi-Fi based backhaul link established between the Intel NUC and HP EliteDesk computers. Second, we deployed a 5G Network to assess potential bottlenecks associated with the design of the proposed solution.

In this test, we deployed the Core Network on the same computer as the gNB, to avoid any bottleneck between these two components, and used a cable-based coaxial connection between a UE and a gNB. Figure 4.1 depicts the test setup used.

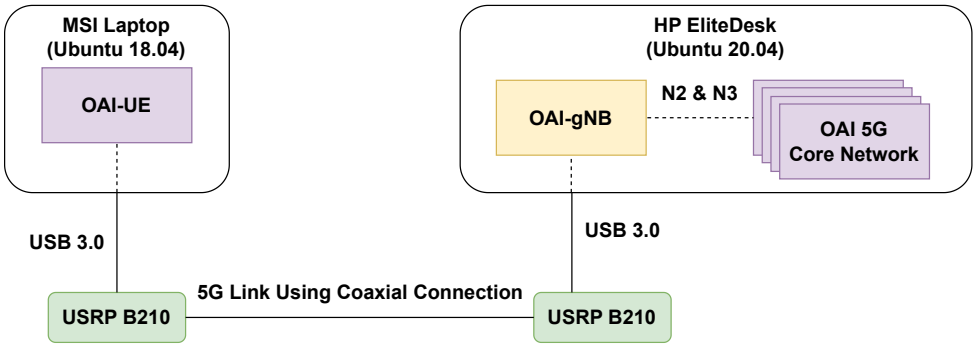

Figure 4.1: Architecture of the deployed scenario, in order to test maximum throughput in the 5G link.

The decision to connect the SDRs via a coaxial cable-based connection instead of antennas was made in order to ensure the best case scenario, avoiding performance losses caused by distance and over-the-air unpredictable interferences. The SDRs connected by a coaxial cable are depicted in Figure 4.2.

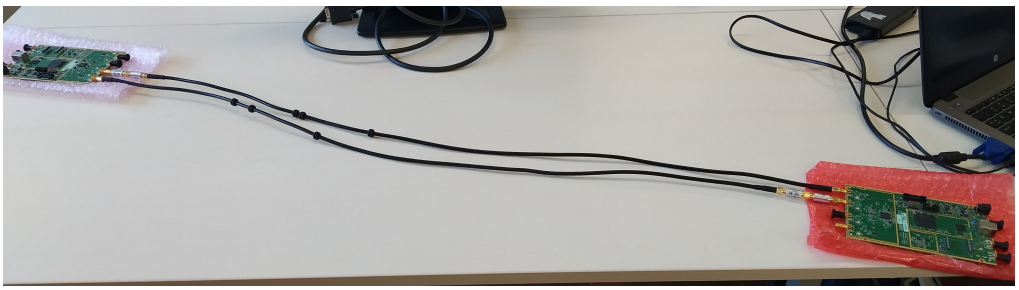

Figure 4.2: SDRs connected using coaxial cables.

The *iPerf* test results between a UE and a Data network using the test scenario in Figure 4.1 are depicted in Table 4.1.

Connection Type	DL (Throughput; Packet Loss Ratio)	UL (Throughput; Packet Loss Ratio)
UDP	24.52 Mbit/s; 2,89%	8.49 Mbit/s; 7.43%
TCP	12.39 Mbit/s; Non-Applicable	8.64 Mbit/s; Non-Applicable

Table 4.1: *iPerf* results between UE and Data Network using a coaxial connection.

The *iPerf* test results for the Wi-Fi link established between the HP EliteDesk (hosting the gNB) and the Intel NUC (hosting the Core Network) are depicted in Table 4.2.

Connection Type	DL (Throughput; Packet Loss Ratio)	UL (Throughput; Packet Loss Ratio)
UDP	34.6 Mbit/s; 2.9%	37.0 Mbit/s; 2.6%
TCP	32.3 Mbit/s; Non-Applicable	31.4 Mbit/s; Non-Applicable

Table 4.2: *iPerf* results for the Wi-Fi based backhaul link established between the Intel NUC and the HP EliteDesk.

The results show the throughput achieved with the Wi-Fi based backhaul link used is higher than the one achieved in the link established between the UE and the gNB using a coaxial cable, with some margin. This means that the Wi-Fi based backhaul link used does not limit the throughput obtained in the access network.

4.3 Core Network

This test is aimed at ensuring that the Core Network was working properly. A 5G Core Network with problems potentially causes most of the other components of the 5G Network to not be able to connect to the 5G Network.

To deploy the OAI 5G Core Network, we deployed the Docker containers by running the command in Figure 4.3 from the folder containing the *docker-compose.yaml*, configured as explained in Section 3.4.1. The *-d* flag was used to run the containers in detached mode, running them in the background without printing the output to the terminal.

```
Core-Network-Host:/Core_network_folder$ docker compose up -d
```

Figure 4.3: Deploying the OAI 5G Core Network, using Docker containers.

After waiting some time for the Docker containers to initialize, we checked their status to ensure that they were working in a healthy state, as shown in Figure 4.4.

```
Core-Network-Host:$ docker ps
(output truncated)
IMAGE                                      STATUS
oaisoftwarealliance/trf-gen-cn5g:latest    Up 52 seconds (healthy)
oaisoftwarealliance/oai-spgwu-tiny:v1.5.0  Up 52 seconds (healthy)
oaisoftwarealliance/oai-smf:v1.5.0         Up 53 seconds (healthy)
oaisoftwarealliance/oai-amf:v1.5.0         Up 53 seconds (healthy)
mysql:8.0                                  Up 55 seconds (healthy)
oaisoftwarealliance/oai-nrf:v1.5.0         Up 54 seconds (healthy)
```

Figure 4.4: Status of the Core Network Docker containers.

Then, we verified that all the containers had IP-based connectivity using the interfaces created in the Host OS (cf. Figure 3.14). For that purpose, we used the *ping* tool from each interface to each Core Network container IP according to Table 3.2; the results are depicted in Figure 4.5.

```
Core-Network-Host:$ ping -I demo-oai 192.168.70.130
64 bytes from 192.168.70.130: icmp_seq=1 ttl=64 time=0.062 ms
Core-Network-Host:$ ping -I demo-oai 192.168.70.131
64 bytes from 192.168.70.131: icmp_seq=1 ttl=64 time=0.065 ms
Core-Network-Host:$ ping -I demo-oai 192.168.70.132
64 bytes from 192.168.70.132: icmp_seq=1 ttl=64 time=0.059 ms
Core-Network-Host:$ ping -I demo-oai 192.168.70.133
64 bytes from 192.168.70.133: icmp_seq=1 ttl=64 time=0.100 ms
Core-Network-Host:$ ping -I demo-oai 192.168.70.134
64 bytes from 192.168.70.134: icmp_seq=1 ttl=64 time=0.054 ms
```

Figure 4.5: Pinging NRF, MySQL Database, AMF, SMF and UPF respectively from Host OS interface (output truncated).

Since all the pings were successful, we conclude that the 5G Core Network was fully operational. An additional check that can be made is to use Wireshark in order to analyze the packets being exchanged in the *demo-oai* interface, which allows for connectivity between the different 5G Docker containers. For illustrative purposes, in Figure 4.6, we can see the PFCP protocol, described in Section 2.2 which is used, in this case, for a heartbeat message exchanged between the SMF and the UPF.

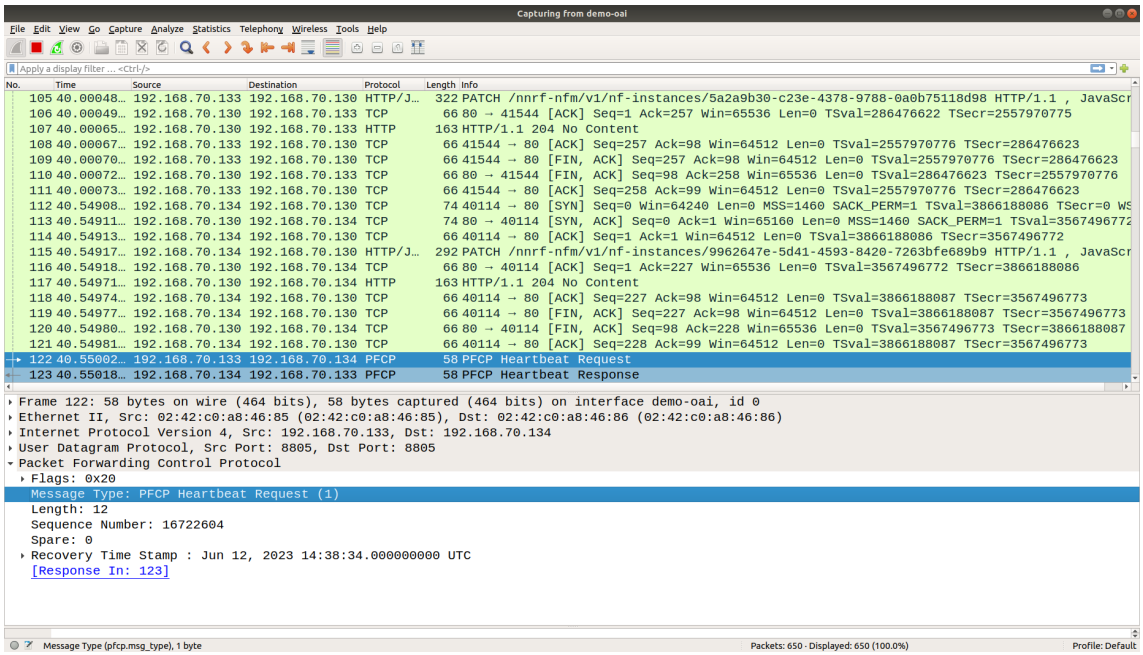

Figure 4.6: 5G Core Network packet exchange, including PFCP Heartbeat messages exchanged between the SMF and the UPF.

4.4 FlexRIC

Regarding the FlexRIC application, when it is deployed, no additional test is necessary beyond confirming the successful launch of the application, as shown in Figure 4.7. When launched, the FlexRIC application remains on standby and waits for any incoming connection requests from an E2 Node, such as the gNB, as depicted in Figure 3.4.

```
Core-Network-Host:/path_to_flexRIC/build/examples/ric$ ./nearRT-RIC
Setting the config -c file to /usr/local/etc/flexric/flexric.conf
Setting path -p for the shared libraries to /usr/local/lib/flexric/
[NEAR-RIC]: nearRT-RIC IP Address = 10.42.0.16, PORT = 36421
[NEAR-RIC]: Initializing
(...)
```

Figure 4.7: Initializing the FlexRIC executable (output truncated).

When the FlexRIC executable is launched without errors, and with the Host OS IP address assigned, then it is fully operational and the gNB can be started.

4.5 gNB

To ensure the correct functioning of the gNB, it should be connected and properly registered in the 5G Core Network. Moreover, the gNB should be connected to the FlexRIC and registered as

an E2 node. Ensuring these connections is important because the connection to the Core Network and the FlexRIC are separate and independent from each other; if one is working as expected, it does not imply that the other is also working correctly.

In order to assess the connectivity between the gNB and the 5G Core Network, we need to check the connection between the gNB Host and the AMF and UPF, as these are the components of the Core Network that gNB needs to communicate with. We tested this by pinging these components from the Host OS of the gNB, as shown in Figure 4.8, after configuring the needed route (cf. Figure 3.17).

```
gNB-Host:$ ping 192.168.70.132
64 bytes from 192.168.70.132: icmp_seq=1 ttl=64 time=8.01 ms
gNB-Host:$ ping 192.168.70.134
64 bytes from 192.168.70.134: icmp_seq=1 ttl=64 time=6.54 ms
```

Figure 4.8: Pinging AMF and UPF, respectively, from gNB Host (output truncated).

After connectivity with the Core Network has been established, the OAI 5G gNB executable, *nr-softmodem*, was executed. The command used to launch the gNB is depicted in Figure 4.9. The *-O* flag indicates that the program should use a configuration file, which is specified by the file's location. The *–gNBs.[0].min_rxtxtime 6* flag specifies that a minimum time for transmitting and reading samples should be used. The *–sa* flag states that the gNB runs in Standalone mode. The *-E* flag specifies that the program should use 3/4 of the sampling rate frequency. Finally, the *–continuous-tx* flag configures the gNB to not disable the transmission even if the reception is activated; according to OAI, this is used to increase stability, because some USRP devices create self-interference when the transmission is disabled [53].

```
gNB-Host:$ sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/CONF
    /gnb.sa.band78.fr1.106PRB.usrpb210.conf --gNBs.[0].min_rxtxtime 6 --sa -E
    --continuous-tx
```

Figure 4.9: Deploying OAI 5G gNB.

After launching the gNB, if the software does not print errors, the gNB starts transmitting and receiving radio signals via the SDR equipment. It is possible to confirm if the SDR USRP B210 board is working correctly by verifying if the transmission and reception Light-Emitting Diodes (LED) light up. If that is the case, as depicted in Figure 4.10, the OAI gNB software is working properly with the SDR.

Figure 4.10: USRP B210 SDR transmitting and receiving radio signals, as indicated by the green and red LEDs, on the top right hand-side.

The fact that the gNB's SDR is working, does not mean that it is correctly integrated on the 5G Network. In order to check if the gNB was successfully registered in the Core Network, we can observe the packets being exchanged, using Wireshark. As shown in Figure 4.11, the gNB successfully registered itself with the AMF via the N2 interface.

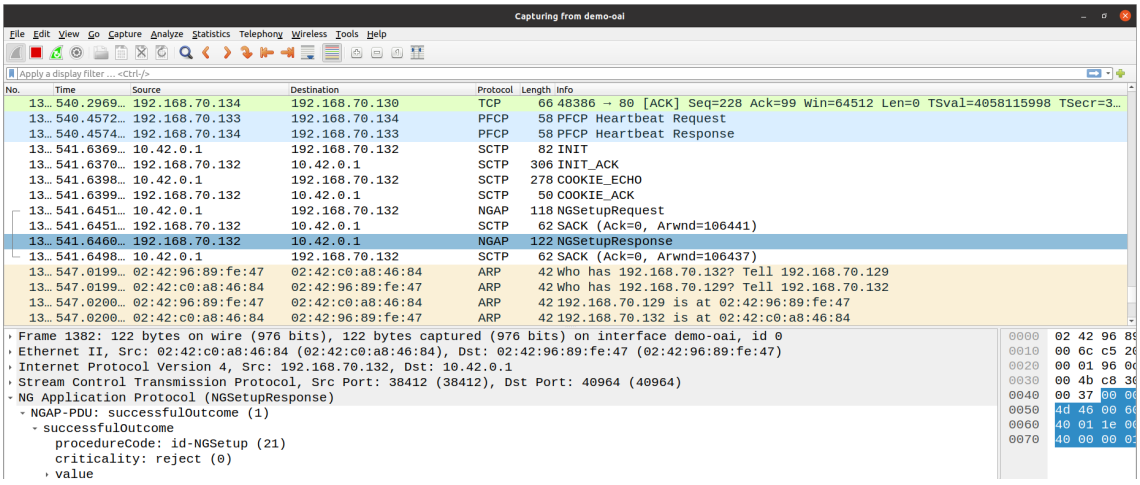

Figure 4.11: gNB successfully registered with the AMF via the N2 interface.

The last test aimed at ensuring that the remaining gNB functions were working as expected, mainly the E2 node. For that purpose, we checked the success of the connection between the gNB's E2 node and FlexRIC. To validate this we observed the output from the terminal where FlexRIC was running and verified that the gNB's E2 was registered with the FlexRIC, as depicted in Figure 4.12.

```
(...)
fd created with 6
Received message with id = 3584, port = 20459
[E2AP] Received SETUP-REQUEST from PLMN 208.95 Node ID 3584 RAN type ngran_gNB
[NEAR-RIC]: Accepting RAN function ID 142 with def = MAC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 143 with def = RLC_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 144 with def = PDCP_STATS_V0
(...)
```

Figure 4.12: gNB's E2 node registering itself with the FlexRIC (output truncated).

We can also verify that the E2 connection between the gNB and the FlexRIC was successful by observing the Wireshark captures between the two components, which are depicted in Figure 4.13.

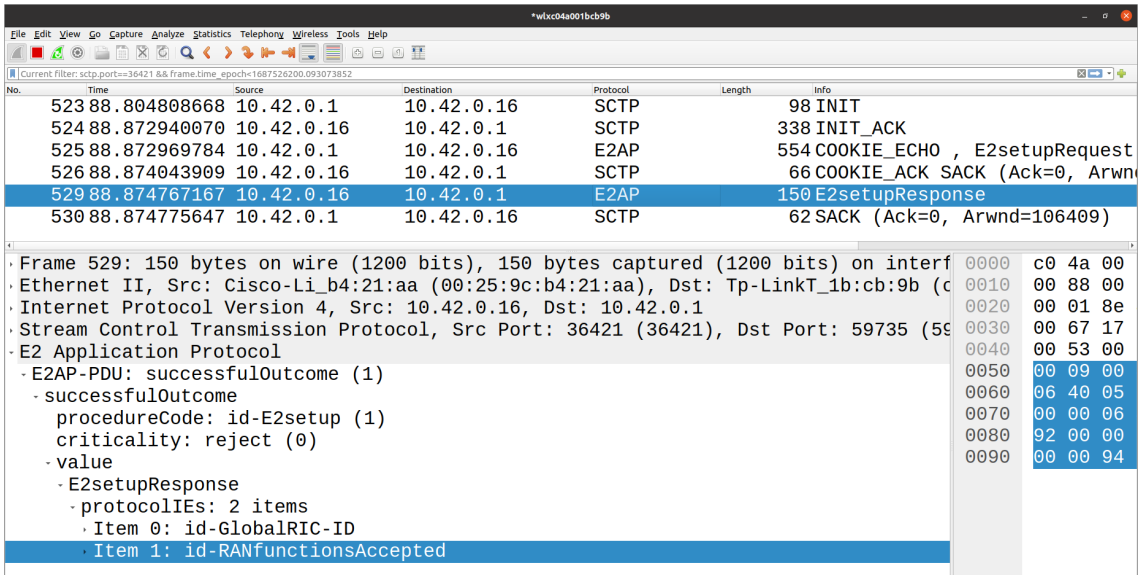

Figure 4.13: Successfull E2 Setup Request and Response between the gNB's E2 node and the FlexRIC.

4.6 UE

To test the proper functioning of the UE we ensured not only its connection to the 5G network but also run performance tests.

Due to the fact that we implemented the OAI 5G UE as a Docker container, we launched and checked its status as we did for the Core Network (cf. Figure 4.3), but from the UE Host computer. If the container is running in a *healthy* state, we need to make sure that the SDR acting as the UE can establish a connection with the SDR connected to the gNB. For this, we placed them in radio range of each other until the LED lights on the UE's SDR lit up in the same way as those on the

gNB, as we demonstrated in Figure 4.10. When the LEDs on both SDRs light up, the SDRs are synchronised and can start exchanging information.

An additional step in order to validate the UE's correct functioning is to check if it is registered with the 5G Core Network; this can be verified by using Wireshark, as depicted in Figure 4.14.

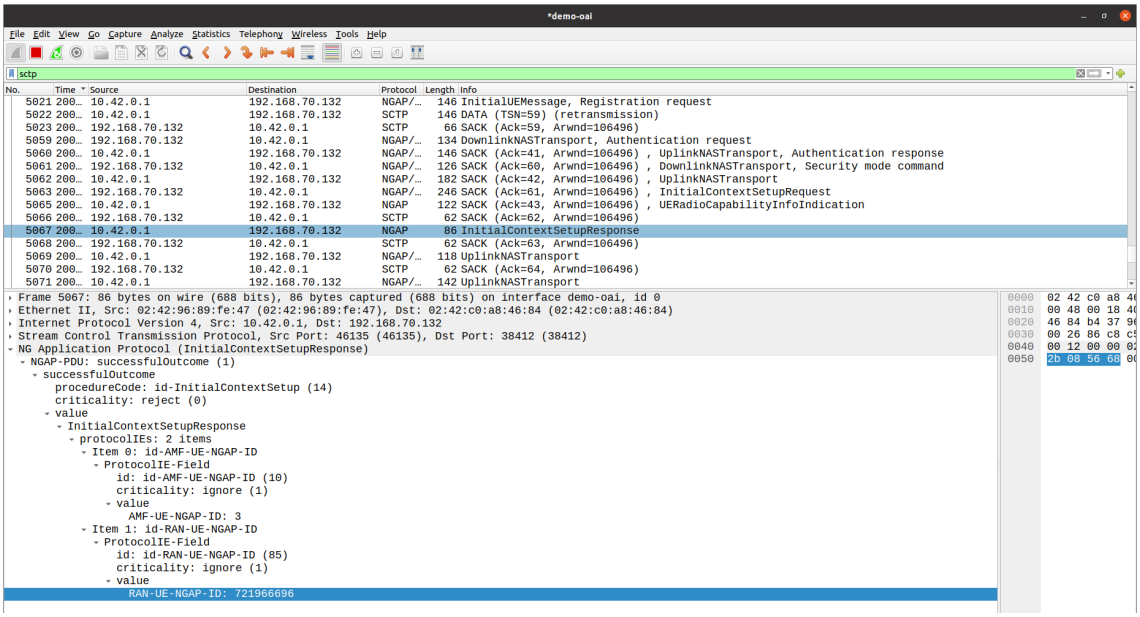

Figure 4.14: Successfull UE registration in the Core Network with AMF.

After the UE is registered in the Core Network, it is assigned an IP address to send and receive IP traffic through the 5G network. This IP address can be checked as depicted in Figure 4.15

```
UE-Host:$ ifconfig
oaitun_ue1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 12.1.1.151  netmask 255.255.255.0  destination 12.1.1.151
```

Figure 4.15: IP TUN interface for the OAI UE created on Host computer (output truncated).

To test if the UE's interface is working properly, we can ping the external DN, as shown in Figure 4.16. If the ping test is successful, this means the UE has connectivity to outside networks using the 5G Network.

```
UE-Host:$ ping –I oaitun_ue1 192.168.71.135
64 bytes from 192.168.71.135: icmp_seq=48 ttl=63 time=17.8 ms
```

Figure 4.16: Ping to external DN from UE (output truncated).

Moreover, we inspected the Core Network using Wireshark. In Figure 4.17, we can observe the GTP protocol being used to encapsulate the Internet Control Message Protocol (ICMP) packets

from the UE to the external DN. The ICMP packets from the UE to DN are encapsulated by the GTP protocol and sent to the UPF, which in turn de-encapsulates them and forwards data traffic to the DN.

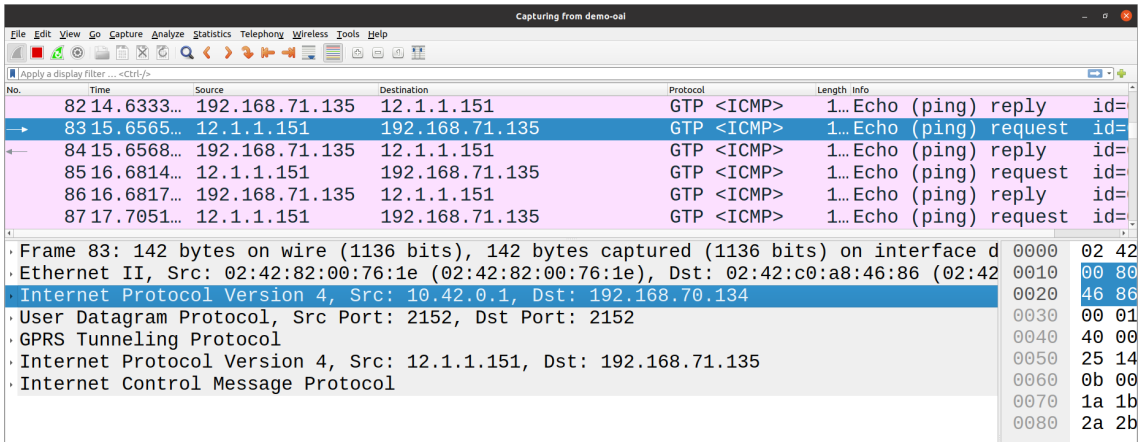

Figure 4.17: ICMP packet, sent from UE to external DN, encapsulated by the GTP protocol.

In order to access the maximum throughput that a UE can achieve using the developed 5G network, we performed both TCP and UDP tests using the *iPerf* tool. The results for throughput and Packet Loss Ratio between the UE and the external DN are depicted in Table 4.3. In the case of the DL tests, the *iPerf* server is launched on the UE and the *iPerf* client is launched on the DN; this order is changed for the UP tests.

Connection Type	DL (Throughput; Packet Loss Ratio)	UL (Throughput; Packet Loss Ratio)
UDP	22.6 Mbit/s; 8.65%	4.2 Mbit/s; 0%
TCP	3.64 Mbit/s; Non-Applicable	4.11 Mbit/s; Non-Applicable

Table 4.3: *iPerf* results between UE and DN in the proposed testbed.

Although the throughput results are low, especially using a TCP connection, they show that there is 5G connectivity ensured by the developed network.

4.7 Mobility Management xApp

After all other components of the 5G Network were deployed and fully operational, we assessed the correct functioning of the developed Mobility Management xApp. In this test, we evaluated if the xApp is able to obtain data from the E2 node on the gNB. For that purpose, the xApp needs to communicate with the FlexRIC software, which allows generating an E2 subscription request and sending it to the E2 node (cf. Figure 3.4): this process can be observed in Figure 4.18.

```
(...)
[iApp]: SUBSCRIPTION_REQUEST xapp_ric_id->ric_id.ran_func_id 142
[E2AP] SUBSCRIPTION REQUEST generated
[NEAR_RIC]: nb_id 3584 port = 30142
(...)
```

Figure 4.18: FlexRIC receiving Subscription Request from the xApp (output truncated).

After the subscription request is made, the E2 node is able to send E2 indication messages with the periodicity defined in our xApp, in this case: 10 ms. In our case, these E2 indication messages contained the near-RT metrics regarding the MAC layer, as it was the service model we subscribed to in the E2 Subscription Request, in order to acquire the values of the SNR, as well as the transmitted and received MAC SDUs. These values are used by the algorithm described in Section 3.2 running in the Mobility Management xApp, in order to determine a suitable position for the Mobile Robotic Platform. The E2 Subscription Request to the E2 RAN node and the following E2 Indication messages are shown in the Wireshark logs in Figure 4.19.

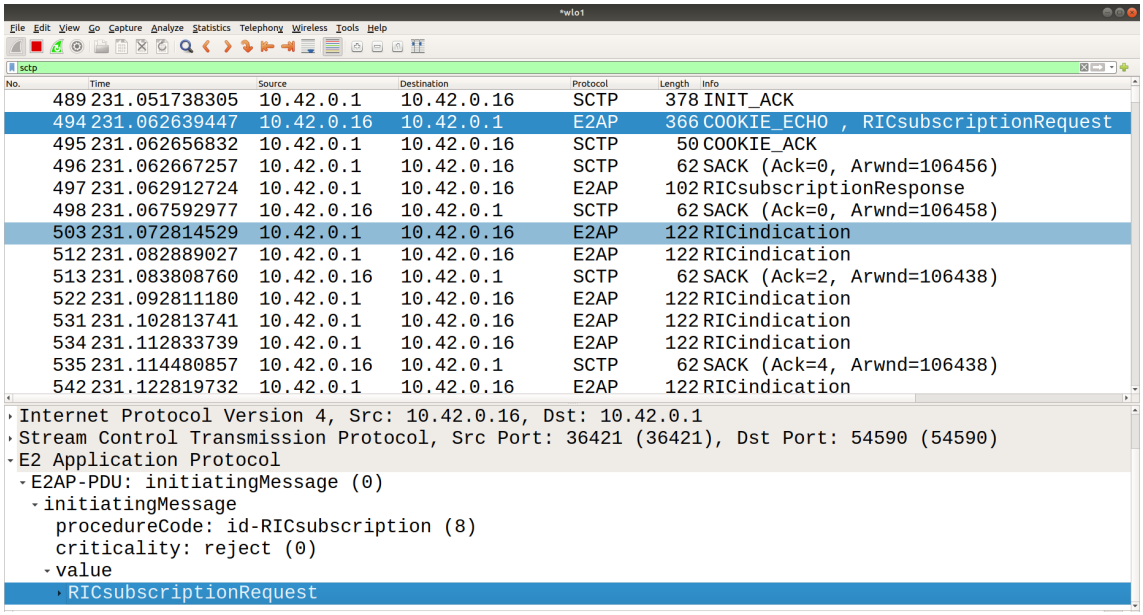

Figure 4.19: E2 Subscription Request and following E2 Indication messages.

The periodicity of 10 ms used for the E2 indication messages is defined within FlexRIC's API, with the other options being 1 ms, 2 ms, and 5 ms. We wanted to test how accurately the xApp can receive these messages within the chosen interval of 10 ms, especially due to the fact that we may need to consider additional delays imposed by the Wi-Fi link. This test is important, because, in our Mobility Management xApp, the way we measure the average SNR over, for example, 1 second, is to calculate the average SNR value from 100 consecutive E2 indication messages (100 $\times$ 10 ms = 1 s)). This means that if there is a significant delay in the E2 indication messages, it can impact the intended timings on the Mobility Management xApp.

To that end, we conducted a test in which we measure the time it takes to receive 100 consecutive E2 indication messages, which should amount to 1 s of total time. This test was conducted for 60 s, allowing us to measure a total of 6000 E2 indication messages. The test results are presented in Table 4.4.

	Value for 100 indication messages	Average value for 1 indication message
Average value	1.023388 s	10.23388 ms
Highest value	1.213530 s	12.13530 ms
Lowest value	0.950583 s	9.50583 ms

Table 4.4: Test results for E2 indication messages, exchanged with a periodicity of 10 ms.

As we can see, using the Wi-Fi link, the expected time of 10 ms between each E2 indication message, defined in FlexRIC's API, is very close to the actual time taken to receive each E2 indication message.

4.8 Use Case Validation

To test the proposed solution implemented in a real system, we carried out a use case testing scenario in an outdoor environment, following the architecture presented in Figure 3.3. For logistical reasons, during this test, we changed the roles of the HP EliteDesk and MSI Laptop computers: the MSI Laptop was used to deploy the gNB and Robotic Control application, and the HP EliteDesk was used to deploy the UE. The deployment of the Core Network, FlexRIC, UE, and SDR board is depicted in Figure 4.20.

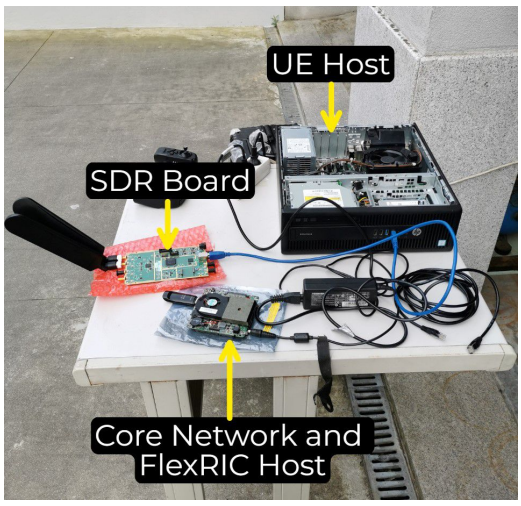

Figure 4.20: Core Network, FlexRIC, UE, and SDR board deployed in the use case testing scenario.

In Figure 4.21, we can see the SDR securely placed on top of the Mobile Robotic Platform.

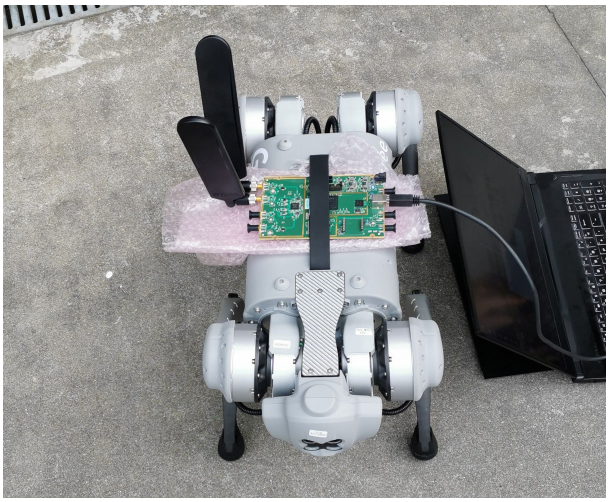

Figure 4.21: The USRP B210 SDR board on top of the Mobile Robotic Platform.

In Figure 4.22, a wide view of the testing scenario used to conduct the tests is depicted.

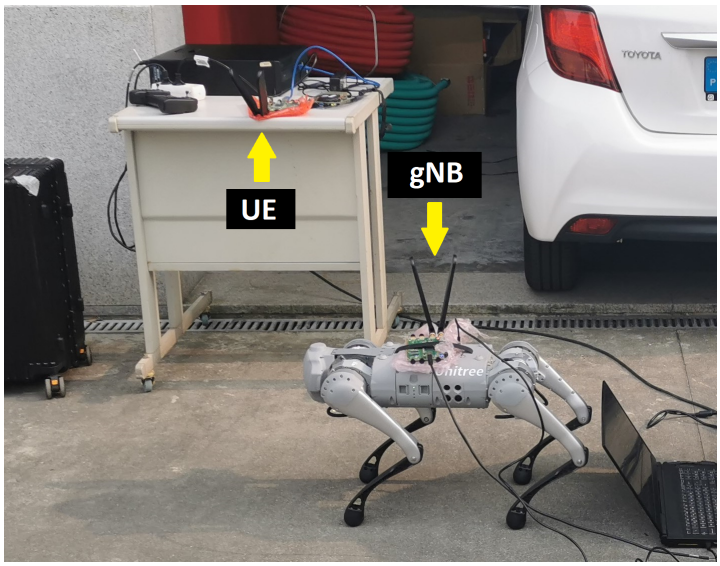

Figure 4.22: Outdoor test scenario.

This test aimed at validating if the developed system worked as a whole, considering end-to-end connectivity between the UE and an external DN, while the positioning of the Mobile RAN was controlled autonomously by the Mobility Management xApp, in order to increase the channel quality, especially in terms of the SNR values. For this, we placed the Mobile RAN approximately one meter away from the SDR board connected to the UE, which was defined as the default position of the Mobile RAN. The Mobile RAN was positioned to move parallel to the UE. After initiating the 5G Network, UE, and Mobility Management xApp, we placed an obstacle between the two SDR boards, which obstructed the line of sight between them: this obstruction made the SNR of the connection drop. The test scenario is depicted in the diagram of Figure 4.23.

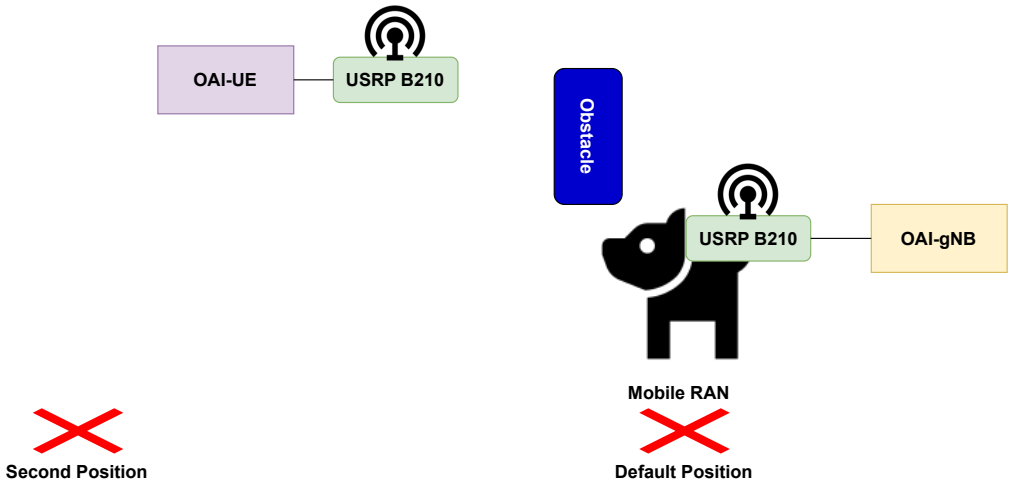

Figure 4.23: Diagram depicting the placement of the Mobile RAN, UE, and obstacle, in order to test the proposed solution.

The test was successful. When the obstacle was introduced between the SDR boards, the Mobility Management xApp detected a reduction in the connection's SNR to a value lower than 30 dB. Then, the xApp sent a control command for the Mobile RAN to move to the second position. In the second position, due to the fact that there were no obstructions between the SDRs, the value of the SNR was above 30 dB, and the Mobility Management xApp instructed the Mobile RAN to stay in the second position.

Besides this, we also conducted another test scenario in which we placed the Mobile RAN in a default position, further away from the UE, as depicted in Figure 4.24.

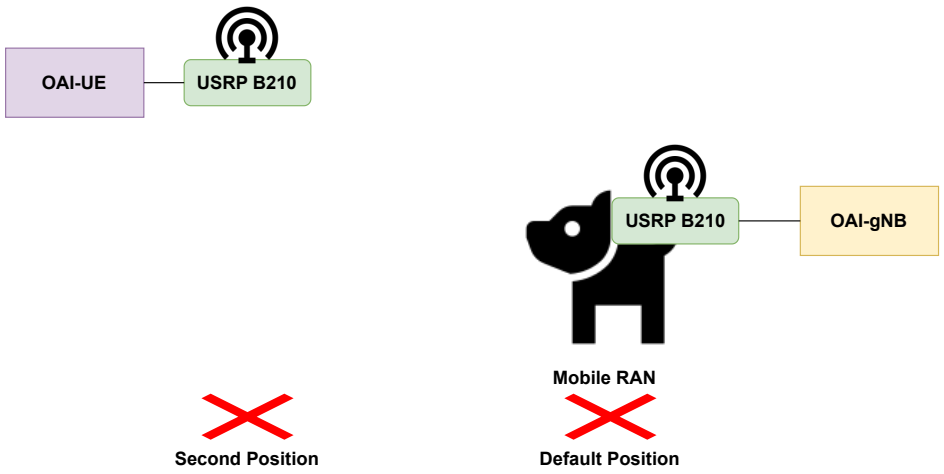

Figure 4.24: Diagram depicting the placement of the Mobile RAN and the UE, in order to test the implemented solution.

This test was also successful. When the Mobility Management xApp was launched, it immediately detected that, in the default position, the value of the SNR was lower than 30 dB, due to

the longer distance between the SDRs. Then, it sent a control command in order for the Mobile RAN to move to the second position, which, being closer to the UE, led to an increase in the SNR value to around 30 dB. Sometimes this value decreased below 30 dB, but since the value was still greater than the value measured on the default position, the Mobility Management xApp instructed the Mobile RAN to stay in the second position.

4.9 Discussion

The tests conducted allowed us to validate the deployed 5G Network and its performance. The objectives of this dissertation were achieved by means of the autonomous control and positioning of a mobile 5G RAN node, which is able to improve the quality of the 5G connection between the UE and the 5G Network in different use case scenarios, such as dynamic obstruction of the line of sight or variable distance between the UE and the gNB.

The use of the O-RAN architecture enabled open interfaces such as E2, which allowed us to obtain the needed metrics from the RAN and use them in the proposed algorithm, employed in our Mobility Management xApp to position the mobile RAN.

The first test performed allowed us to ensure that the Wi-Fi based backhaul link does not hinder the performance results. The subsequent tests allowed us to validate each component of the 5G Network, both individually and regarding their correct interoperation. Moreover, they allowed us to assess the performance achieved with the proposed solution. The final test allowed us to validate the proposed solution in a real-world environment while ensuring that the Mobility Management xApp is able to autonomously reposition the mobile RAN in order to enhance the 5G connection quality.

The experimental evaluation carried out allowed us to conclude that the proposed solution is a valuable contribution when it comes to the implementation of mobility management functions in a real-world 5G Network. By taking advantage of the O-RAN architecture, which provides the flexibility to integrate novel xApps in the 5G network, a user can easily develop custom xApps to manage and monitor the network, without the need to modify the individual components or protocol implementations in the RAN.

Chapter 5

Conclusion

5.1 Conclusions

The goal of this dissertation was to develop a private 5G Standalone RAN, based on the O-RAN architecture, composed of a RAN node deployed on a mobile robotic platform, as well as a solution to autonomously control the placement of the RAN by taking advantage of the benefits of the O-RAN architecture. The proposed solution allowed us to overcome some of the limitations in commercial network deployments.

On the one hand, the O-RAN architecture plays a crucial role in improving overall access and interoperability in the RAN, by providing open and standardized interfaces. These interfaces, such as the E2 interface, allow operators to control and extract metrics from the RAN via simple and easy to develop xApps, without having to rely on closed proprietary systems. On the other hand, the use of mobile RAN nodes, which can be dynamically controlled according to the users' requirements and environment, allows 5G Networks to offer better QoS and QoE in multiple scenarios. The ability to autonomously place the RAN nodes provides higher flexibility compared to static network deployments, allowing the network to enhance the management of dynamic conditions, such as unexpected bursts of traffic, dynamic obstacles, and variable distances between the UEs and the RAN.

The proposed solution takes advantage of both the O-RAN architecture, and the concept of mobile networks, and implements a 5G SA network architecture based on a mobile RAN, which is placed and carried by a Mobile Robotic Platform. Due to the mobile nature of the RAN, our solution implements a wireless-based backhaul connection in order for the gNB to establish communications with the 5G Core Network and the Near-RT RIC. The open-source software used to implement the 5G Core Network and RAN was Open Air Interface, while to implement the Near-RT RIC we relied on the FlexRIC software. In order to enable 5G communications between the UE and gNB, the hardware used was the USRP B210 SDR board. Finally, in order to autonomously control the placement of the mobile RAN we developed a Mobility Management xApp.

The proposed Mobility Management xApp is able to collect metrics from the RAN, namely the SNR values of the 5G connection, and compute control actions in order to better place the

Mobile Robotic Platform, increasing the connection quality between gNB and UE.

Throughout the development of the proposed solution, we faced a number of implementation challenges. The first one was the time spent in order to get a functional 5G Network using OAI's software. OAI is very complex in terms of code and implementation, making it difficult to have a 5G Network working properly, because problems would arise even if just one configuration field or flag is missing or not properly configured. This is exacerbated by the fact that some of its documentation lacks organization, and there is a general lack of certain pieces of information. Even though the documentation present on the GitLab page of OAI has been updated multiple times throughout the development of this dissertation, improving along the way, there are still numerous problems with it, ranging from contradicting information on how to properly setup components of the 5G Network, to complete lack of information regarding the necessity and purpose of certain flags needed for the proper functioning of the software. Besides this, OAI software is also very demanding in terms of hardware requirements, making the software hard to run and leading to occasional crashes and errors.

Similarly to OAI, the documentation on FlexRIC's Gitlab page is also very scarce. The main problem in developing the proposed xApp was the fact that there was no documentation regarding the metrics that can be extracted from the RAN and the corresponding variables within the FlexRIC API. As such, we had to extensively analyze and understand most of the code base in order to locate the *structs* that contain the required information. Despite being much less resource intensive and more stable than OAI's 5G software, there are some scenarios that can cause the FlexRIC software to crash.

The SDRs used also posed an additional challenge to the implementation of the proposed solution. This is mainly due to the problems related to sending/receiving samples to/from the SDRs. This can be justified by many reasons, but especially due to the use of a USB connection. During the execution of the software running on the 5G network nodes connected to the SDRs, sometimes the software stopped receiving samples from the SDR device. Even relaunching the software yielded the same result, and the only way to fix this problem was to unplug and replug the USB connection between the SDR and the Host Computer. This behaviour caused the software to crash or return errors.

The main challenges mentioned and other minor difficulties made it arduous to test the system as a whole, due to time constraints, as well as stability problems: even if a single system component malfunctioned or had stability issues, it would force us to stop testing and relaunch the whole testbed. In spite of that, we were able to conduct the necessary tests in order to validate the proposed solution. These tests included validating each component individually and also their correct interoperability. Moreover, a test scenario using a real-world use case was carried out, in order to validate the system as a whole

Although there are several improvements that can be made to the proposed solution, the objectives of this dissertation were fully achieved, resulting in three main contributions: 1) a Private 5G SA Network, based on the O-RAN architecture, deploying a mobile RAN node that can be dynamically positioned for increased performance; 2) a Mobility Management xApp, capable of

autonomously controlling the placement of the mobile RAN, by analyzing the SNR values and other metrics collected from the RAN, in order to provide better 5G connectivity to a UE, and 3) a baseline testbed implementation, which can be adapted to test other mobility management functions and scenarios, including solutions based on ML.

5.2 Known Limitations and Future Work

Even though the present dissertation achieved all the proposed objectives, there are still some system limitations and room for improvements.

The first limitation is related to the movement of the Mobile Robotic Platform. Currently, it is not aware of any environmental information such as terrain hazards and obstacles, moving blindly according to the controls of the Mobility Management xApp. One important improvement would be to integrate a system using computer vision and sensing information, in order to provide the Mobile Robotic Platform with information about its surroundings, preventing it from causing or sustaining any type of damage. This information would also allow for the development of more complex Mobility Management xApps.

The second limitation pertains to the computing units and SDR models used to implement the proposed solution: the USRP B210 boards. Although being one of the SDR models recommended by OAI, this model has limited performance compared to the other recommended models. Besides that, the computing units used to run alongside this SDR model did not meet the hardware requirements listed by OAI. Moreover, these boards also caused stability problems, mainly due to the USB connection established with the Host computer, which caused problems transmitting and receiving samples, even when no other USB devices were connected to the Host computer. In future works, the use of more powerful computing units and more advanced USRP models employing Ethernet-based fronthaul connections, such as the X310, N300, N310, N320, and X410 is recommended.

The third limitation is related to the use of the Wi-Fi link between gNB and Core Network. In our case, and as demonstrated with the test in Section 4.2, the Wi-Fi connection used ensured enough capacity to meet the throughput requirements associated with the 5G RAN. With better equipment, the performance enabled by the 5G technology is potentially orders of magnitude higher, and a Wi-Fi connection may not be suitable to act as a link between gNB and Core Network, especially in environments with other networks sharing the medium. One possible alternative is to use Integrated Access and Backhaul (IAB)[3]. Besides exploring RAN splitting into CUs and DUs, IAB enables wireless connectivity between a fixed base station, acting as an IAB-donor, and a mobile IAB-node, through the NR Uu interface, making the entire connection from UE to Core Network only rely on the 5G technology. This may be considered as part of future work.

Appendix A

Functional Testing

This appendix is used to present some tests that were performed during the development of this dissertation; they helped us to reach the final system that was used to deploy the proposed solution (cf. Figure 3.3). These tests mainly helped us to eliminate deployment options that were initially considered, developed, deployed, and tested but ended up not being used due to performance and stability issues.

A.1 Not Deploying the gNB on the Intel NUC

The main design choice that needs to be addressed herein is the use of the Intel NUC to deploy the Core Network instead of the gNB, as it would be a better fit for the latter – since the Intel NUC is a mini-computer, it would easily be carried by the Mobile Robotic Platform alongside the SDR board. This restricted us to using the HP EliteDesk as the Host for the gNB, which it cannot be placed on the Mobile Robotic Platform. For this reason, we placed the USRP B210 SDR on-board the Mobile Robotic Platform, connected to the HP EliteDesk via USB 3.0. Despite this inconvenience, this setup allowed us to validate the proposed solution as the radio unit that propagates and receives the 5G radio signal is still mobile. From the point of view of the UE and Core Network, there is no difference between our setup, and one where the entire RAN (gNB Host computer + USRP B210 radio unit) is placed on-board the Mobile Robotic Platform.

This design choice was made due to the lack of computational power available in the Intel NUC to deploy the OAI gNB. As shown in Figure A.1, during the runtime of the OAI gNB, the program prints the character *L* multiple consecutive times. This happened especially when the network was loaded with traffic exchanged between the UE and the Data Network. This character is printed when the gNB software is unable to read a sample quickly enough from the SDR device, leading to the loss of samples and communications errors.

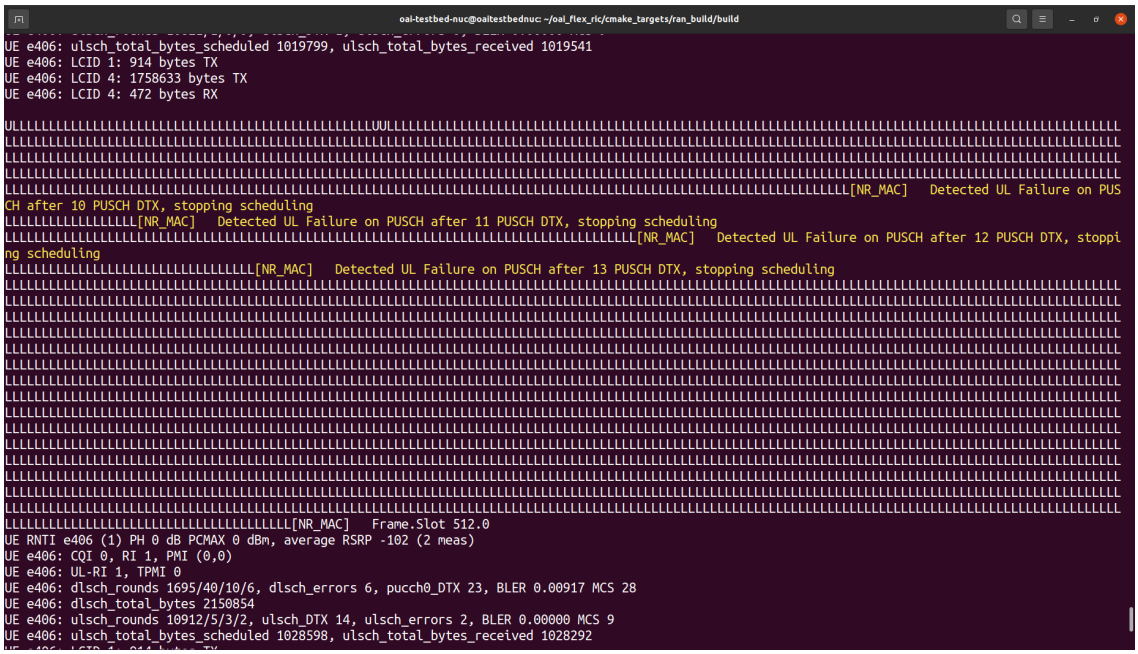

Figure A.1: gNB running in OAI NUC, which is unable to read samples from the SDR fast enough.

These performance issues were very rare with the HP EliteDesk while occurring almost constantly in the Intel NUC, leading to successive communications errors, including when it comes to control signalling messages.This implies disruption between UE and gNB, causing the UE to be deregistered from the 5G Network, as shown in Figure A.2.

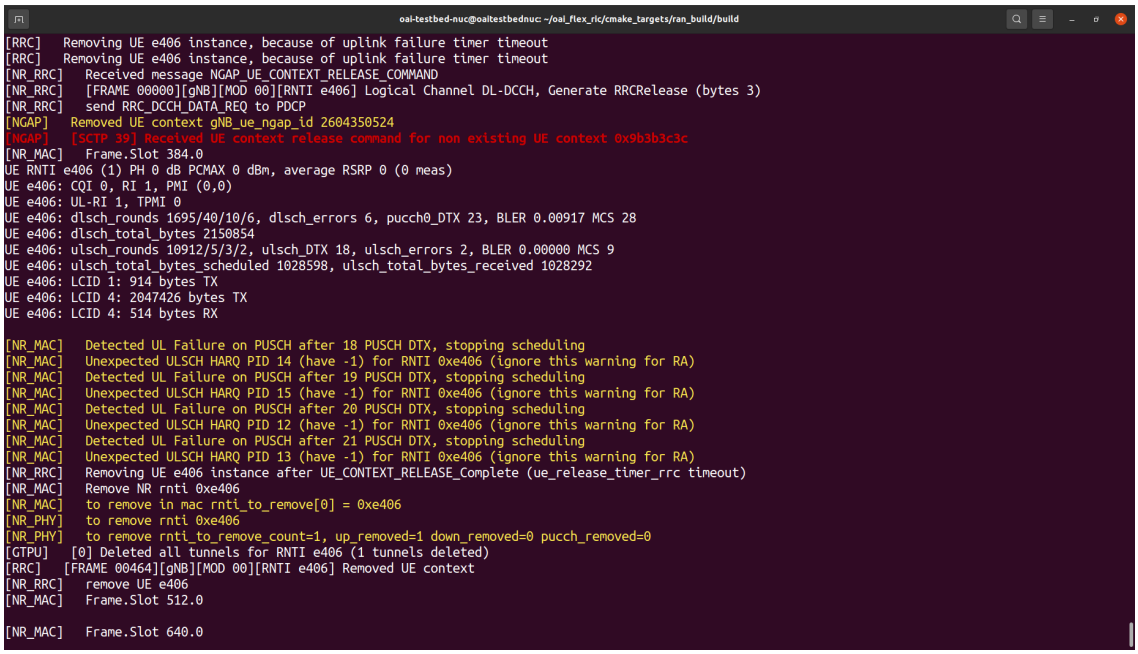

Figure A.2: UE being deregistered from the Core Network, due to the Intel NUC performance limitations.

This setup made it impossible to test the UE connection to the 5G network under load and led us to rely on the HP EliteDesk to host the gNB instead.

A.2 Not Splitting the gNB into CU and DU

As explained in Section 3.3.1, one of the main reasons for choosing OAI over srsRAN to deploy the 5G RAN was the fact that it allowed for RAN splitting, by separating the gNB into a CU and DU. This deployment option was highly preferred because it makes the proposed solution more compliant with the O-RAN architecture; ultimately, we decided not to use it for the final proposed solution.

Initially, after installing the FLexRIC patch on the OAI software, when we tried to launch the *nr-softmodem* software to deploy the gNB, with the option to deploy it as a CU or DU, the program yielded an error, related to the ID of the CU or DU node passed to the library that implements the E2 node. When the OAI gNB is launched in CU or DU mode, its RAN ID variable, used in the code, is set to *0*, but the E2 library implemented by FlexRIC expects a value different than *0*, so the program returns with an error. After some changes to the source code, we were able to develop a small patch, which made it so that the RAN ID variable was different than *0*, and the software executed as expected, for both CU and DU deployments.

After being launched, both nodes worked as expected, the CU registering itself with the Core Network, the DU registering itself with the CU, and both of them registering as E2 nodes with the FlexRIC. However, whenever we launched an xApp, after the E2 subscription request reached the CU, the software implementing the CU crashed with a segmentation fault. At this point, it became clear that the FlexRIC software was not developed and tested considering OAI's RAN split option.

Despite this setback, and due to the fact that the only metrics that were needed for the Mobility Management xApp pertained to the MAC layer and can be obtained from the DU, we decided to deploy a RAN based on a regular OAI CU and a DU implementing the FlexRIC's patch and our own patch. This deployment approach is depicted in Figure A.3.

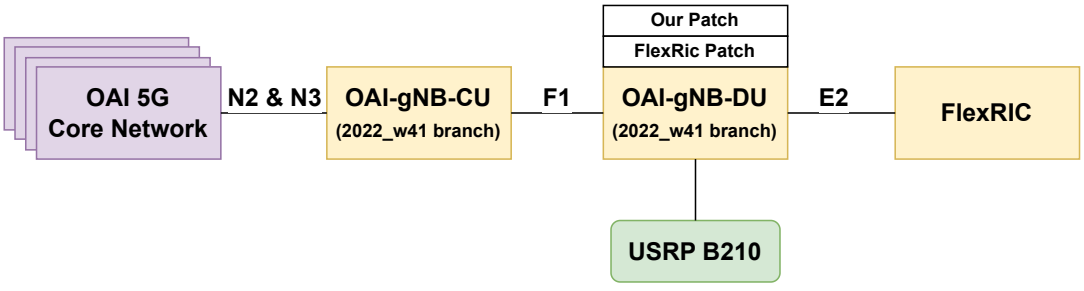

Figure A.3: Deployment approach in which the gNB is split into CU and DU.

Even though this deployment worked successfully, there was one major issue with it. Sometimes, the Random Access Channel (RACH) Procedure, which takes place when a UE tries to first connect to a gNB triggered failures, causing the software executing the DU to crash.

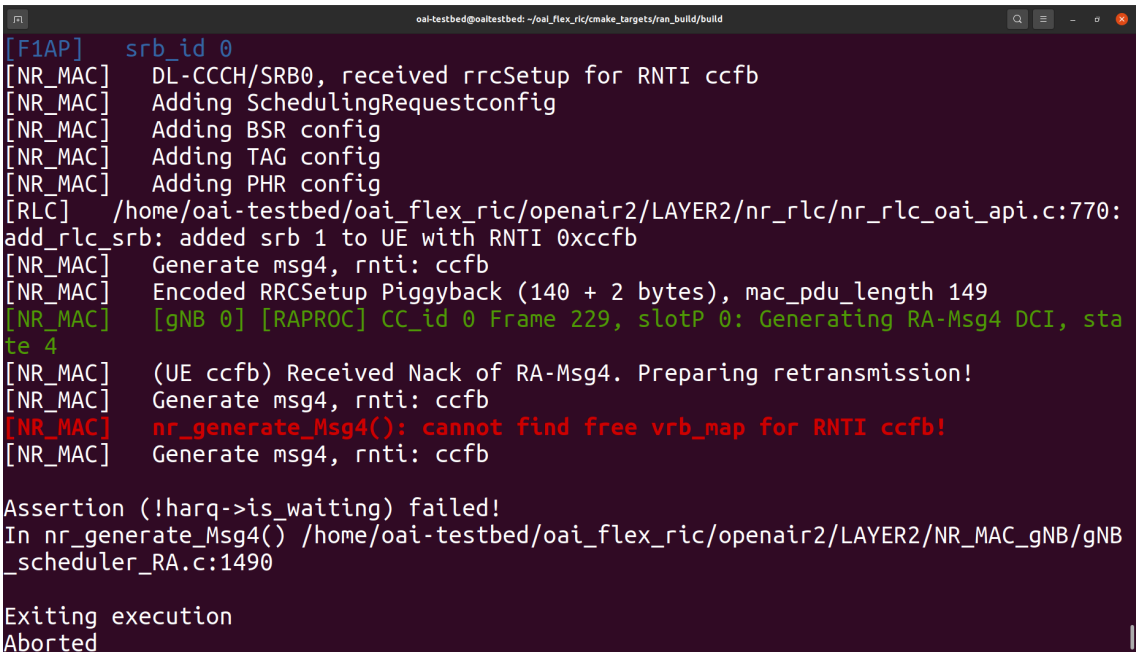

Figure A.4: DU software crashing after UE initiated RACH Procedure.

This problem seemed to happen at random, approximately 80% of the time. Also, this problem never occurred when executing the gNB as a single unit. Due to this, the already high instability of the system when using a full gNB, and due to the limited time to test the system, we decided not to split the RAN in our proposed solution, as it would make it even more difficult to test and validate the proposed solution considering this new point of failure.

References

[1] IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. Recomendation ITU-R M.2083, September 2015.

[2] System architecture for the 5G System (5GS), 3GPP TS 23.501 V16.14.0, September 2022.

[3] NR; NR and NG-RAN Overall description; Stage-2, 3GPP TS 38.300 V16.10.0, September 2022.

[4] NG-RAN; NG general aspects and principles , 3GPP TS 38.410 V16.4.0, October 2021.

[5] Technical Specification Group Core Network and Terminals; Interface between the Control Plane and the User Plane Nodes , 3GPP TS 29.244 V16.11.0, March 2023.

[6] Faisal. C-RAN vs Cloud RAN vs vRAN vs O-RAN- A simple Guide. Available: https://telcocloudbridge.com/blog/c-ran-vs-cloud-ran-vs-vran-vs-o-ran/.

[7] Michael Wang. 5G, C-RAN, and the Required Technology Breakthrough. Available: https://medium.com/@miccowang/5g-c-ran-and-the-required-technology-breakthrough-a1b2babf774, June 2018.

[8] NG-RAN; Architecture description, 3GPP TS 38.401 V16.10.0, September 2022.

[9] O-RAN Architecture Description, O-RAN.WG1.O-RAN-Architecture-Description-v06.00, March 2022.

[10] O-RAN Control, User and Synchronization Plane Specification 9.0, O-RAN.WG4.CUS.0-v09.00, July 2022.

[11] O-RAN A1 interface: General Aspects and Principles 3.0, O-RAN.WG2.A1GAP-v03.00, October 2022.

[12] O-RAN Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles 2.02, O-RAN.WG3.E2GAP-v02.02, July 2022.

[13] O-RAN Control, User and Synchronization Plane Specification 10.0, O-RAN.WG4.CUS.0-v10.00, October 2022.

[14] FlexRIC Wiki. Available: https://gitlab.eurecom.fr/mosaic5g/flexric/-/wikis/Create-a-xApp.

[15] Unitree Go 1 Product Page. Available: https://shop.unitree.com/products/unitreeyushutechnologydog-artificial-intelligence-companion-bionic-companion-intelligent-robot-go1-quadruped-robot-dog?variant42363559641321.

[16] O-RAN Alliance. Available: https://www.o-ran.org/.

[17] Shunliang Zhang. An Overview of Network Slicing for 5G. *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111-117, June 2019. doi:10.1109/MWC.2019.1800234.

[18] 3GPP. Available: https://www.3gpp.org/about-us.

[19] ONF White Paper. Software-Defined Networking-The New Norm for Networks. April 2012.

[20] Erik Dahlman, Stefan Parkvall, and Johan Skold. *5G NR: The Next Generation Wireless Access Technology*. Academic Press, Inc., USA, 1st edition, 2018.

[21] Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3, 3GPP TS 24.501 version 16.12.0, June 2022.

[22] NG-RAN; NG Application Protocol (NGAP) , 3GPP TS 38.413 V16.11.0, September 2022.

[23] General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U), 3GPP TS 29.281 V16.2.0, March 2021.

[24] O-RAN Alliance, Who we are. Available: https://www.o-ran.org/who-we-are.

[25] O-RAN Use Cases and Deployment Scenarios, O-RAN White Paper, February 2022.

[26] ETSI. Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges. October 2012.

[27] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. February 2022. doi:https://doi.org/10.48550/arXiv.2202.01032.

[28] O-RAN Deployment Scenarios and Base Station Classes 3.0, O-RAN.WG7.DSC.0-v03.00, October 2021.

[29] O-RAN Management Plane Specification 10.0, O-RAN.WG4.MP.0-v10.00, October 2022.

[30] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms, December 2021. URL: https://arxiv.org/abs/2112.09559, doi:10.48550/ARXIV.2112.09559.

[31] Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. SCOPE: An Open and Softwarized Prototyping Platform for NextG Systems. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '21, page 415–426, New York, NY, USA, June 2021. Association for Computing Machinery. URL: https://doi.org/10.1145/3458864.3466863, doi:10.1145/3458864.3466863.

[32] David Johnson, Dustin Maas, and Jacobus Van Der Merwe. NexRAN: Closed-Loop RAN Slicing in POWDER -A Top-to-Bottom Open-Source Open-RAN Use Case. In *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, WiNTECH'21, page 17–23, New York, NY, USA, January 2021. Association for Computing Machinery. URL: https://doi.org/10.1145/3477086.3480842, doi:10.1145/3477086.3480842.

[33] Pratheek S. Upadhyaya, Aly S. Abdalla, Vuk Marojevic, Jeffrey H. Reed, and Vijay K. Shah. Prototyping Next-Generation O-RAN Research Testbeds with SDRs, May 2022. URL: https://arxiv.org/abs/2205.13178, doi:10.48550/ARXIV.2205.13178.

[34] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. FlexRIC: An SDK for next-generation SD-RANs. In ACM, editor, *CONEXT 2021, 17th International Conference on Emerging Networking EXperiments and Technologies, 7-10 December 2021, Munich, Germany (Virtual Conference)*, Munich, 2021. doi:http://doi.org/10.1145/3485983.3494870.

[35] Ahan Kak, Van-Quan Pham, Huu-Trung Thieu, and Nakjung Choi. ProSLICE: An Open RAN-based approach to Programmable RAN Slicing. In *2022 IEEE GLOBECOM, IEEE Global Communications Conference, 4–8 December 2022, Rio de Janeiro, Brazil, Hybrid: In-Person and Virtual Conference, Accelerating the Digital Transformation through Smart Communications*, Rio de Janeiro, December 2022.

[36] David Maia. Control and Positioning of a 5G Radio Access Node Deployed in a Mobile Robotic Platform. Master's thesis, Faculdade de Engenharia da Universidade do Porto, July 2022.

[37] Open Air Interface. Available: https://openairinterface.org/.

[38] srsRAN Project. Available: https://www.srslte.com/.

[39] OAI Project. Available: https://openairinterface.org/oai-5g-ran-project/.

[40] srsRAN O-RAN gNB Overview. Available: https://docs.srsran.com/projects/project/en/latest/knowledge_base/source/oran_gnb/source/index.html.

[41] OSC Near-RT RIC Documentation. Available: https://docs.o-ran-sc.org/projects/o-ran-sc-it-dep/en/latest/index.html.

[42] FlexRIC Repository. Available: https://gitlab.eurecom.fr/mosaic5g/flexric.

[43] OSC's Near-RT RIC Installation and Requirements. Available: https://openaicellular.github.io/oaic/oran_installation.html.

[44] Intel NUC Board NUC5i5MYBE Specifications. Available: https://www.intel.com/content/www/us/en/products/sku/84862/intel-nuc-board-nuc5i5mybe/specifications.html.

[45] USRP B210 Product Page. Available: https://www.ettus.com/all-products/ub210-kit/.

[46] USRP X310 Product Page. Available: https://www.ettus.com/all-products/x310-kit/.

[47] OAI System Requirements. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/OpenAirSystemRequirements.

[48] W5084k Dipole Antenna Datasheet. Available: https://www.farnell.com/datasheets/3812050.pdf.

[49] OAI Repository. Available: https://gitlab.eurecom.fr/oai/openairinterface5g.

[50] UHD Wiki. Available: https://kb.ettus.com/UHD.

[51] Universal TUN/TAP device driver Documentation. Available: https://www.kernel.org/doc/html/v5.8/networking/tuntap.html.

[52] iPerf Website. Available: https://iperf.fr/.

[53] Merge Request addressing the addition of the *–continuous-tx* flag to increase stability. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/merge_requests/1439.